Nouvelles contributions du boosting en apprentissage automatique

Henri-Maxime Suchier

École doctorale de Saint-Étienne

Nouvelles Contributions du Boosting en Apprentissage Automatique

Thèse présentée à l'Université Jean Monnet de Saint-Étienne
pour obtenir le grade de :

Docteur de l'université Jean Monnet de Saint-Étienne
Mention : Informatique.

par Henri-Maxime Suchier

Eurise Faculté des Sciences et Techniques

Soutenue le 21 Juin 2006, devant le jury composé de :

M. Pierre Dupont	Professeur des Universités	Rapporteur
M. Jean-Daniel Zucker	Professeur des Universités	Rapporteur
M. François Jacquenet	Professeur des Universités	Examinateur
M. Marc Sebban	Professeur des Universités	Directeur
M. Jean-Christophe Janodet	Maître de Conférences	co-Directeur

à Sonia

Remerciements

Je tiens tout d'abord à remercier Pierre Dupont et Jean-Daniel Zucker pour avoir accepté de rapporter cette thèse, ainsi que François Jacquenet, qui a accepté d'en être l'examinateur.

Il est à peu près certain que cette thèse n'aurait pas aboutit sans mes encadrants qui, par leur motivation sans faille, et leurs encouragements (notamment au cours de ces derniers mois), m'ont apporté un soutien considérable durant mon doctorat. Je remercie donc Marc Sebban et Jean Christophe Janodet pour avoir accepté d'être mes directeurs de thèse. Je souhaite pouvoir continuer à avancer dans le monde de la recherche en travaillant dans le respect des valeurs qu'ils m'ont transmises.

Il semblerait d'ailleurs que ce soit les mêmes valeurs que partage l'ensemble des membres du laboratoire EURISE. Les divers séminaires, réunions et groupes de travail auxquels j'ai pu assister au sein du laboratoire ont été très riches en découvertes. Que tous les acteurs de ces évènements soient ici remerciés.

Quant à l'ambiance conviviale qui règne au laboratoire... les pauses café, au sujets de discussions parfois improbables, en sont le meilleur exemple. Elles feront également partie des bons souvenirs que je garderai de mon passage à l'EURISE. Une pensée, donc, aux amateurs de café[1] du labo.

J'ai eu la chance de pouvoir participer durant ma thèse à deux écoles d'été. De par ce que j'ai pu y apprendre, et de par les gens avec qui j'ai pu y discuter, je pense que ces quelques semaines ont largement contribué à l'élaboration de ma culture scientifique. Je remercie donc Colin de la Higuera, directeur du laboratoire, pour m'avoir poussé à y assister, et plus généralement pour m'avoir permis de participer à divers évènements scientifiques, aussi bien nationaux qu'internationaux.

Les fluctuations du moral font partie intégrante de la vie d'un doctorant. Quand rien ne va bien, rien de tel que la famille et les amis pour vous aider à remonter la pente. Que toutes les personnes qui ont réussi à me changer les idées pendant les instants de doute, et qui ont de fait largement contribué à l'aboutissement de cette thèse, soient ici remerciés. Il serait impossible d'en dresser une

[1] et à défaut, de thé ;-)

liste complète, aussi, pour ne froisser personne, me permettrais-je de n'en citer qu'une : Julie, ma soeurette.

Enfin, au moment de terminer ces remerciements, toutes mes pensées se tournent vers Sonia, sans qui la vie n'aurait pas une telle saveur.

Table des matières

Introduction

Introduction

L'apprentissage automatique est un domaine émergent de recherche en informatique dont le champ d'application n'a cessé de croître ces dernières années. Historiquement, il a trouvé ses sources dans de nombreux domaines, comme l'informatique théorique (calculabilité, complexité, théorie des langages), l'intelligence artificielle, les statistiques, les sciences cognitives, et plus récemment, l'image, le traitement automatique des langues, ou encore la biologie.

Quel que soit son domaine d'application, un processus d'apprentissage automatique exploite des données (numériques ou symboliques) décrivant selon un ensemble d'attributs un concept cible inconnu, et utilise une méthode d'apprentissage permettant d'inférer un modèle de la réalité (appelée hypothèse), que l'on souhaite généralisable à de nouvelles observations.

Dans le cadre de l'apprentissage *non supervisé*, la tâche consistera en la découverte de similarités entre les observations, dans le but de regrouper celles-ci en sous-ensembles, appelés *clusters* ou *classes*. Par exemple, dans le domaine du *Web Intelligence*, on s'intéressera à regrouper des utilisateurs du web en sous-ensembles, à partir de l'observation de leurs habitudes de navigation. Ce regroupement permettra de mettre en évidence un certain nombre de profils d'utilisateurs.

Dans le cadre de l'apprentissage *supervisé*, qui constituera le contexte de cette thèse, les observations sont accompagnées d'une information complémentaire relative à leur appartenance ou non au concept. On parlera alors d'apprentissage à partir d'*exemples* (positifs) si seulement des observations du concept sont fournies à la méthode d'apprentissage, et d'apprentissage à partir d'*exemples positifs* et *négatifs*, si des contre-exemples sont également disponibles. Notons que lorsque le concept cible présente plusieurs modalités (par exemple, la famille d'un champignon), on parlera d'apprentissage multi-classes, et les observations servant à l'apprentissage devront alors décrire, par leur étiquette, ces différentes catégories.

Enfin, on a assisté ces dernières années à l'émergence de l'apprentissage *semi-supervisé* dont le but est d'exploiter la grande masse d'exemples naturellement non étiquetés (plus facilement accessibles) et de choisir judicieusement un petit

nombre d'exemples étiquetés pour simplifier en temps et en coûts le travail de l'apprentissage supervisé.

Selon le cadre d'apprentissage utilisé, la mise en oeuvre du processus d'inférence d'hypothèses diffère fortement. Le travail pionnier de Gold (1967) cherchait à apprendre automatiquement des langages. Les motivations de ces recherches ne provenaient pas des applications pratiques possibles mais plutôt des questions théoriques proches de la calculabilité. Quelle famille des concepts peut-on apprendre ? Sous quelles hypothèses ? etc. S'inspirant probablement des sciences cognitives, Gold imagine le processus d'apprentissage de la manière suivante : l'algorithme reçoit un flot infini de données qui caractérisent l'intégralité du concept à apprendre, à la limite, tout comme un individu "reçoit" des phrases relatives à sa langue maternelle tout au long de sa vie ; cet algorithme génère un nouveau modèle des données à chaque fois qu'il en reçoit une nouvelle, comme un individu affine sa connaissance de la langue française au fil du temps. Pour Gold, avoir appris, c'est trouver le concept qui a généré les données après avoir vu un nombre fini (potentiellement grand) d'observations. Lorsque c'est possible, c'est à dire lorsque les conditions d'apprenabilité du concept sont remplies, on dit alors que l'algorithme a convergé.

Le contexte d'apprentissage initié par Gold a donné lieu à de nombreux résultats positifs et négatifs d'apprenabilité en fonction des familles de langage ciblées. Par exemple, les langages réguliers ne sont pas apprenables à partir d'exemples positifs seuls, mais ils le sont quand des exemples positifs et négatifs sont fournis. De plus, le modèle de Gold, auquel on reproche souvent d'être simpliste, s'est affiné au fur et à mesure pour tenir compte de contraintes de temps par exemple (modèles de Gold (1978), de Pitt (1989) ou de de la Higuera (1997)).

D'autres cadres d'apprenabilité sont ensuite apparus et ont fait l'objet de nombreux travaux, comme celui proposé par Angluin (1987a) ; Angluin (1987b). Ce modèle mime un jeu de devinette : il fait intervenir deux entités, un oracle qui sait tout d'un certain concept à apprendre et un apprenant dont l'objectif est de trouver ("deviner") le concept de l'oracle. L'apprentissage est alors un jeu de questions/réponses entre les deux entités, qu'on appelle des requêtes, qui peuvent être de natures très diverses (requêtes d'appartenance, d'équivalence, statistiques, etc.) et dont on cherche souvent à minimiser le nombre. Le jeu s'arrête quand l'apprenant a trouvé le concept cible. Angluin a ainsi pu montrer que son modèle permettait d'apprendre les langages réguliers avec un nombre polynomial de requêtes d'appartenance et d'équivalence.

Il est à noter que les deux cadres précédents relèvent de l'inférence exacte : on cherche à trouver exactement "le" concept cible. On peut remarquer, bien que cette contrainte puisse avoir des applications pratiques (en vérification de circuit, par exemple), qu'elle n'est pas très adaptée à de nombreuses situations réelles où

chercher le concept cible n'a aucun sens.

Considérons, par exemple, la démarche d'un physicien ou d'un biologiste. Par ses observations du monde réel, par ses nombreuses expériences, il collecte un grand nombre de données dont il va tirer un modèle, une équation différentielle par exemple. Il attend de ce modèle plusieurs caractéristiques, comme le fait qu'il modélise correctement les phénomènes observés (pouvoir de généralisation), qu'il se comprenne au vu des théories actuelles (intelligibilité), etc. A t-il pour autant trouvé le concept cible? Si l'on considère les modèles de la lumière, par exemple, deux théories longtemps contradictoires l'ont expliquée soit de manière ondulatoire, soit de manière particulaire, sans qu'il soit possible de "préférer" l'une à l'autre. On voit bien ici qu'un modèle unique n'a pas de sens. Ce n'est d'ailleurs pas ce que cherche ce scientifique : il sait que quelle que soit son équation, elle aura des failles, et devra toujours être affinée jusqu'à ce qu'elle soit abandonnée au profit d'une nouvelle équation, plus conforme aux connaissances en mathématiques, en physique ou en biologie d'une nouvelle époque, plus en phase donc avec le paradigme courant.

Le cadre proposé par Valiant (1984), dans lequel se situe cette thèse, nous semble donc bien plus pertinent que les modèles d'apprentissage précédents. Il suppose *a priori* que le concept cible aura du mal a être identifié, et se fixe donc comme objectif de trouver la meilleure hypothèse possible, ayant un pouvoir prédictif le plus grand possible, assurant un écart faible avec le concept cible. On parlera alors de *PAC* apprenabilité, pour *Probably Approximately Correct*. Il n'est pas inutile de rappeler que la construction d'une hypothèse peut avoir plusieurs finalités. D'une part, on pourra s'intéresser à ses propriétés en termes d'intelligibilité, c'est à dire qu'elle sera utilisée pour expliquer le concept qu'elle modélise (*qu'est-ce qui fait que l'animal que j'observe est une oie ?*). D'autre part, l'hypothèse pourra être considérée en termes de performances de prédiction, c'est à dire qu'elle sera employée pour prédire de nouvelles observations (*quel est ce nouvel animal ?*). A ce titre, la notion d'*erreur en généralisation*, qui désignera la probabilité d'erreur de l'hypothèse inférée, sera considérée avec intérêt, et l'un des objectifs principaux de l'apprentissage dans le cadre de Valiant sera de la minimiser.

Un domaine émergent de l'apprentissage supervisé est celui des méthodes ensemblistes. Ici, les hypothèses considérées sont en fait composées d'un ensemble d'hypothèses dites *de base*. Suite à notre remarque précédente, il apparaît clairement que la composition d'un tel ensemble d'hypothèses est nuisible à l'intelligibilité de l'hypothèse résultante. C'est donc l'aspect prédictif des hypothèses dites *globales* ainsi construites qui est privilégié dans ce contexte, par rapport à leur intelligibilité. Pour différencier les méthodes ensemblistes entre elles, on s'intéresse en général soit au processus de construction de chacune des hypothèses (quelles

méthodes d'apprentissages sont appliquées? quels exemples sont utilisés?), soit à la stratégie de combinaison de leurs réponses dans une phase de prédiction.

Le boosting, issu des travaux de Freund et Schapire il y a une quinzaine d'années, est une des méthodes ensemblistes les plus réputées aujourd'hui. La raison de cet engouement pour le boosting vient probablement du fait qu'il a fait l'objet d'une quantité remarquable de travaux, tant sur ses propriétés théoriques que sur ses performances en pratique. Avec le boosting, les hypothèses sont construites séquentiellement par une même méthode d'apprentissage, appelée *apprenant faible*. Chaque nouvelle construction utilise une nouvelle distribution sur l'échantillon d'apprentissage, dans laquelle sont favorisés les exemples mal appris par les hypothèses précédemment construites. Il s'agit donc d'exemples difficiles à apprendre, et l'idée de concentrer l'apprentissage sur eux semble pertinente. Par exemple, avant une période d'examens, lorsqu'un étudiant reprend ses cours durant la semaine de révision, il va très probablement au fil des jours se concentrer sur les parties du programme qu'il a du mal à assimiler, et passer ainsi plus d'heures dessus. La stratégie du boosting repose sur ce principe très intuitif qu'une telle méthode de révision devrait être payante en fin de semaine. Au cours de la dernière décennie, cette intuition a été confirmée par nombre de travaux, non seulement expérimentaux, mais également théoriques.

Cependant, certaines difficultés liées à l'utilisation du boosting dans des applications réelles demeurent aujourd'hui encore des problèmes ouverts, et c'est pour cette raison que cette méthode a constitué le thème central de cette thèse. Les problème actuels du boosting sont liés au fait que ses principaux résultats théoriques, mais c'est également le cas pour ceux des modèles d'apprenabilité comme le cadre *PAC*, sont développés dans un contexte "aseptisé", faisant fi de la nature particulière des données du monde réel. Les nouvelles technologies d'acquisition de données, comme le web, ont permis ces dernières années de composer de grandes bases de données pouvant être exploitées par les techniques d'apprentissage automatique. Cependant, au moins trois types de difficultés sont rencontrées par le boosting face à ces données réelles.

Premièrement, de par sa stratégie de repondération, le boosting est intrinsèquement sensible à la présence de données d'apprentissage bruitées. En commettant des erreurs sur de tels exemples, l'apprenant faible favorise l'augmentation de leur poids et modifie donc à tort la distribution statistique des exemples d'apprentissage, forçant ainsi les hypothèses ultérieures à apprendre le bruit. On se trouve alors dans une situation où l'hypothèse globale *sur-apprend* : l'hypothèse produite s'écarte du concept cible, et ses capacités de prédiction diminuent sensiblement. Historiquement, ce problème est probablement celui qui a activé le plus tôt la communauté du boosting, avec notamment l'apparition de l'algorithme BrownBoost de Freund (1999). Bon nombre de travaux ont ensuite suivi, mais

comme nous le verrons par la suite, sans apporter de solution idéale au problème. La première contribution majeure que nous apportons dans cette thèse a pour objectif de proposer un nouveau cadre théorique de boosting permettant la gestion de données bruitées.

Deuxièmement, la séparabilité parfaite des sous-classes d'un concept étant rares dans le monde réel, de forts chevauchements de distributions statistiques apparaissent dès lors que l'on travaille sur des applications concrètes. En d'autres termes, un grand nombre de situations réelles présentent des exemples et contre-exemples partageant exactement la même représentation. Il est ainsi impossible de caractériser le sexe d'un individu à la seule vue de son prénom, *Claude*, *Stéphane*, ou encore *Dominique* étant portés aussi bien par des hommes que des femmes. Plus formellement, ces chevauchements de distributions correspondent à l'erreur bayésienne qu'aucune méthode d'apprentissage ne pourra réduire. Mais dans le cas du boosting, une forte erreur bayésienne va plus loin qu'une simple augmentation de la borne basse de l'erreur théorique atteignable. Elle a en fait un impact direct sur la vitesse de convergence du boosting comme ont pu le montrer Nock et Sebban (2001). Dans ce contexte, nous allons proposer une stratégie permettant de détecter rapidement de tels exemples situés en région bayésienne, augmentant ainsi considérablement la vitesse de convergence vers l'erreur optimale.

Enfin, le boosting est confronté à une dernière difficulté liée à la nature même des nouvelles données que nous manipulons. Alors que pendant longtemps les bases de données sont restées homogènes, c'est à dire contenant des informations de même type (numérique ou symbolique), les bases de données modernes présentent fréquemment des attributs hétérogènes, comme du texte, des images, du son, des informations numériques, etc. Alors qu'il existe des méthodes d'apprentissage efficaces pour chaque type d'attributs, le boosting n'offre pas de cadre théorique permettant de faire cohabiter de manière coopérative durant l'apprentissage plusieurs méthodes d'inférence. Dans le cadre de cette thèse, et c'est ce qui consistera notre dernière contribution majeure, nous proposons d'aborder ce problème en proposant un nouveau schéma de boosting faisant intervenir les prédictions de l'ensemble des apprenants faibles courants.

Notons dès à présent que cette thèse est avant tout théorique fournissant un ensemble de résultats fondamentaux dans le domaine du boosting. Néanmoins, et puisque ceux-ci ont été développés pour répondre aux problèmes liés à la nature des données réelles, nous avons toujours eu le soucis dans ce manuscrit d'effectuer de larges études expérimentales, ce qui a nécessité un travail de développement important.

L'organisation générale de cette thèse est la suivante. Dans une première partie, nous commencerons par donner des notions générales de théorie de l'apprentissage, avant de dresser un tour d'horizon des principales méthodes ensemblistes. Le boosting est ensuite introduit en détail, puis les principaux résultats théoriques le concernant sont présentés.

Dans une deuxième partie, après avoir mis en évidence les problèmes du boosting face aux données bruitées, nous présentons et détaillons le modèle mis en place afin de rendre le boosting tolérant aux données corrompues ou non pertinentes. Les résultats théoriques obtenus dans ce cadre sont ensuite présentés. Puis nous mettons ce modèle en pratique, tout d'abord sur des données de type numériques, puis dans un second temps sur des données symboliques, plus précisément dans le cadre de l'inférence grammaticale.

Une dernière partie présente le modèle de boosting appliqué à des données constituées d'attributs de natures hétérogènes, faisant collaborer un pool d'apprenants faibles afin d'exploiter au mieux l'information contenue dans chacun des attributs. Là encore, les performances du modèle proposé sont établies théoriquement, puis un ensemble d'expérimentations sont présentées afin de montrer l'efficacité de notre méthode de boosting.

Première partie

Apprentissage et Méthodes Ensemblistes

1 Théorie de l'Apprentissage Automatique

Résumé : Ce chapitre présente le cadre de l'apprentissage automatique dans lequel se situe cette thèse et détaille les problèmes qui y sont rencontrés, notamment celui du dilemne bias/variance. Nous rappelons les principaux résultats théoriques dans le cadre PAC avant d'introduire les méthodes ensemblistes.

1.1 Contexte de l'apprentissage automatique

L'objectif de l'apprentissage automatique est de modéliser un concept cible, défini sur une population, à partir d'observations d'individus de celle-ci. A partir de la nature de ces observations, trois types d'apprentissage peuvent être effectués. Le premier correspond au cas où les observations ne décrivent pas explicitement le concept. L'existence de ce dernier sera donc supposée, et il s'agira de le décrire de la manière la plus vraisemblable, sous forme de classes, en exploitant des mesures de similarité. On parlera alors d'*apprentissage non supervisé*. Le deuxième s'effectue lorsque les observations décrivent cette fois de manière explicite le concept. Celles-ci en guident la modélisation. On parlera alors d'*apprentissage supervisé*. Notons dès à présent que l'ensemble des contributions présentées dans cette thèse se positionne dans cette seconde catégorie. Enfin, une dernière situation correspond au cas où on dispose, en plus d'un petit nombre d'observations décrivant le concept, d'une grande quantité d'observations non classées. Le but est d'exploiter celle-ci afin d'améliorer la modélisation du concept durant l'apprentissage supervisé. On parlera dans ce cas d'*apprentissage semi-supervisé*.

Ce chapitre a pour objectif de présenter le scénario particulier de l'apprentissage supervisé dans lequel cette thèse se place, et de définir les notions qui y seront fréquemment manipulées. Illustrons l'ensemble de ces définitions par un

exemple.

Considérons la situation où un expert de la cueillette de champignons accepte d'initier un novice à son art. Il emmène alors son apprenti lors de son excursion hebdomadaire, durant laquelle il lui nomme chaque champignon rencontré. Malgré son niveau élevé d'expertise, il se montre toutefois incapable de commenter ses affirmations, la verbalisation totale de sa connaissance étant délicate. Le novice décide alors de prendre des notes : pour chaque champignon, il consigne la couleur et la forme du chapeau, ainsi que la couleur du pied. De retour chez lui, il lui faut, s'il désire être capable de reconnaître seul les différentes espèces, formuler des règles permettant d'identifier un champignon. Celles-ci exprimeront un lien entre les caractéristiques observées et l'espèce du champignon. Elles pourront par exemple permettre d'identifier de manière fiable les noms des champignons rencontrés, ou bien de discriminer les spécimens comestibles de ceux qui ne le sont pas.

Détaillons maintenant de manière plus formelle les spécificités de cette situation. Les entités en présence sont les suivantes :

- *Une population d'individus* Ω, dont la taille est potentiellement infinie. Chaque individu ω de Ω est décrit sous forme d'un vecteur x de p attributs : $x = \langle a_1, \ldots, a_p \rangle$. Dans notre illustration, le cueilleur considère la population Ω de champignons selon les attributs a_1 : la *couleur*, a_2 : la *forme du chapeau*, et a_3 : la *couleur du pied*. Chaque individu ω est donc décrit ici comme un point dans un espace de représentation à trois dimensions.
- *Un concept*, ou problème d'apprentissage, sur Ω. Il s'agit en fait d'une fonction f qui à chaque individu $\omega \in \Omega$ associe une étiquette parmi un ensemble Y de labels possibles. f induit ainsi une partition en $|Y|$ sous-ensembles, ou *classes*, sur Ω. La fonction f est ainsi appelée *concept cible* du problème d'apprentissage. Le cueilleur de champignons pourra par exemple considérer le concept cible *variété des champignons*.
- *Un oracle*, qui fournira l'étiquette d'un nombre fini d'individus de Ω. En associant au vecteur de représentation x_i d'un individu, son étiquette y_i, l'oracle permet alors de constituer un ensemble *d'exemples* e_i appelé échantillon d'apprentissage $E = \{ei = (x_i, y_i) : i \in 1..m\}$. Les noms des champignons cités par l'expert (l'oracle de cette situation) associés à la description qu'en fait son apprenti lors de leur excursion, constituent donc, dans notre illustration, l'échantillon d'apprentissage. Un vocabulaire particulier est employé dans le cas des problèmes dits *binaires*, pour lesquels $|Y| = 2$. En effet, une des deux classes regroupera les individus dits *caractéristiques* du concept, dits aussi exemples *positifs*, tandis que la seconde regroupera les individus dits *non caractéristiques*, ou exemples négatifs (ou encore *contre-exemples*). Dans notre illustration, le concept considéré dans

un cas binaire pourrait être celui de la comestibilité des champignons. Les spécimens propres à la consommation en seraient les individus caractéristiques, et les vénéneux, ceux non caractéristiques.

1.2 Modélisation d'un concept

Le but de l'apprentissage est donc de construire, à partir de l'échantillon d'apprentissage E, un modèle du concept f, qui devra être capable, non seulement de retrouver les étiquettes des exemples de E, mais également de prédire correctement le plus souvent possible l'étiquette des autres individus de Ω. En d'autres termes, le modèle construit sur l'échantillon d'apprentissage devra être généralisable à Ω. Par la suite, nous utiliserons le terme d'*hypothèse* h pour faire référence à un tel modèle, qui partitionnera l'espace de représentation par des *frontières de décision*. Le terme de *méthode d'apprentissage* désignera toute technique de production d'une telle hypothèse (voir Fig. 1.1). Dans le cas du cueilleur de cham-

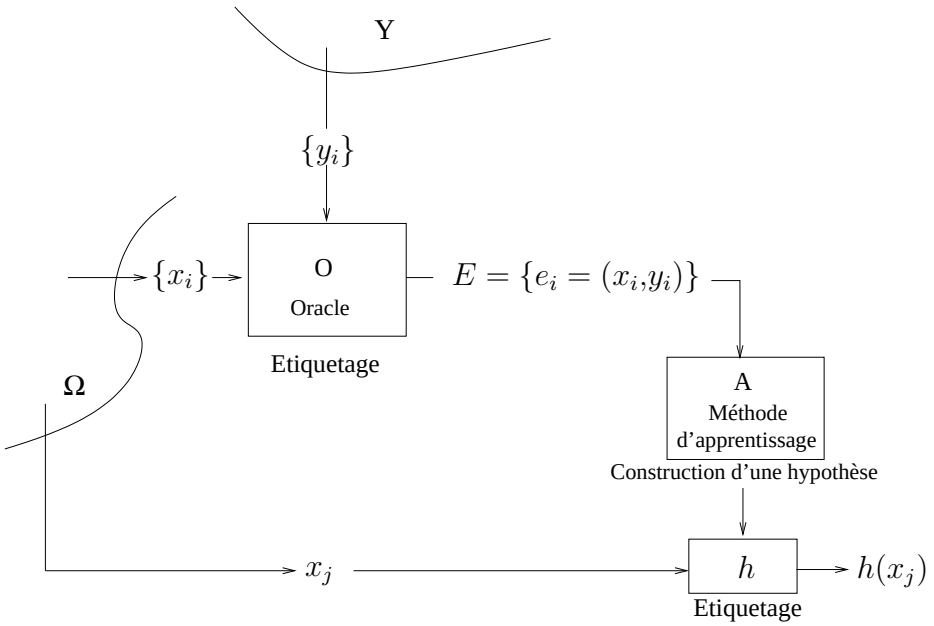


FIG. 1.1 – *Le problème de l'apprentissage automatique : étant donné un échantillon d'apprentissage* $E = \{e_i = (x_i,y_i) : i \in 1..m\}$*, formés par des individus de* Ω *étiquetés par un oracle* O*, le but d'un méthode d'apprentissage* A *est de construire une hypothèse* h *à partir de* E*, généralisable à* Ω*.*

pignons novice, une hypothèse pourra par exemple prendre la forme d'un arbre de décision (Breiman et al., 1984), qui pourrait être celui de la Fig. 1.2. Un tel

modèle partitionne l'espace de représentation en procédant par tests successifs sur les attributs. Ainsi, un champignon dont le chapeau est de couleur brun et de forme sphérique, et dont le pied est de couleur brune, est un bolet bronzé. Chaque feuille de l'arbre correspond à un sous-ensemble de l'espace de représentation des exemples, et se voit associer la classe unique, ou éventuellement majoritaire, des exemples situés dans ce sous-ensemble.

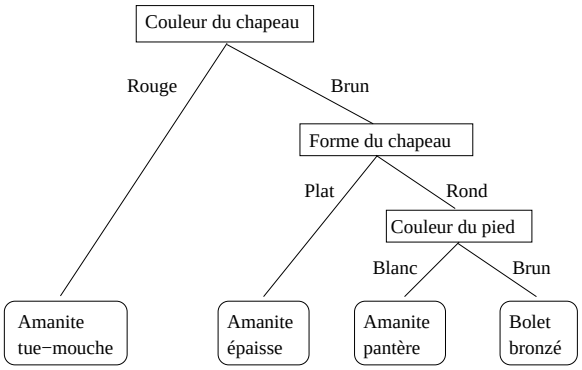


Fig. 1.2 – *Exemple d'arbre de décision sur le problème du cueilleur de champignons, construit sur les attributs "forme du chapeau", "couleur du chapeau" et "couleur du pied". L'étiquetage d'un nouvel individu s'opère en effectuant les tests successifs correspondant aux noeuds de l'arbre (à partir de la racine), et en attribuant l'étiquette associée (éventuellement par majorité) à la feuille rencontrée après le dernier test.*

1.3 Principes inductifs et compromis biais/variance

S'il est souhaitable d'obtenir une hypothèse h modélisant exactement le concept cible f, ceci n'est généralement pas possible en pratique. La plupart du temps, en effet, l'hypothèse h construite n'est pas suffisamment performante pour représenter le concept f dans toute sa richesse. Par exemple, un arbre de décision à un seul niveau de profondeur (appelé aussi *stump*) ne sera certainement pas une hypothèse adaptée pour modéliser le problème de la cueillette de champignons comestibles. Il se peut également que l'ensemble des attributs $\langle a_1, \ldots, a_p \rangle$ décrivant chaque individu soit incomplet et/ou non pertinent pour modéliser le concept. Dans ce cas de figure, situation malheureusement la plus courante dans les applications du monde réel, comment choisir l'hypothèse la plus pertinente ? En d'autres termes, quel principe inductif choisir durant l'apprentissage pour assurer de bonnes propriétés à l'hypothèse que nous serons amenés à retenir ?

Le principe inductif probablement le plus courant en apprentissage automatique est celui de la minimisation du risque empirique (ou principe ERM, pour *Empirical Risk Minimization*), consistant intuitivement à penser qu'en minimisant une fonction d'erreur sur l'échantillon d'apprentissage E (*l'erreur empirique*) on induira alors une hypothèse ayant un bon comportement en généralisation sur Ω, donc une erreur, dite *réelle*, faible. C'est ce principe inductif que nous utiliserons tout au long de cette thèse.

Il est néanmoins à noter que d'autres principes inductifs existent. Par exemple, citons celui de *Minimum Description Length* (MDL), qui préconise de choisir l'hypothèse codant l'information de l'échantillon d'apprentissage E de manière la plus simple et la plus concise possible ; citons également le principe de *Maximum Likelihood* (maximum de vraisemblance), qui suppose l'existence d'une distribution de probabilité sur la famille des hypothèses $\mathcal{H}$ à partir de laquelle h est construite, et qui préconise de choisir l'hypothèse la plus vraisemblable en termes de probabilités au vu de E.

Quel que soit le principe inductif choisi, résoudre un problème d'apprentissage nécessite ensuite de se donner une famille d'hypothèses $\mathcal{H}$ dans laquelle la recherche doit s'effectuer (par exemple, la famille des arbres de décision pour résoudre le problème de la comestibilité des champignons). Ce choix d'une famille d'hypothèses est souvent difficile, dans la mesure où $\mathcal{H}$ doit être en *adéquation* avec le concept cible f à apprendre, concept qui par essence est inconnu (inutile de l'apprendre, sinon !). Cette adéquation peut être exprimée en fonction de deux notions : le *biais* et la *variance* (cf. Fig. 1.3).

Le *biais* est l'erreur liée au fait qu'une famille d'hypothèses choisie $\mathcal{H}$ peut ne pas être capable d'assurer l'apprentissage correct de la cible f appartenant à une famille de concepts $\mathcal{F}$. En d'autres termes, rien ne permet *a priori* d'assurer l'égalité entre l'ensemble des concepts cibles et la famille $\mathcal{H}$. Si on se limite à nouveau à la famille $\mathcal{H}$ de *stumps*, il sera probablement impossible de séparer les champignons comestibles, des non comestibles. La richesse de cette famille $\mathcal{H}$, et donc celle des hypothèses h pouvant être induites durant l'apprentissage, n'est donc pas suffisante pour modéliser le concept cible : le biais sera donc ici grand. Inversement, plus $\mathcal{H}$ sera riche (prenons par exemple les arbres de décision de profondeur n, avec n très grand), plus il sera possible d'induire une hypothèse capable d'apprendre un grand nombre de concepts différents. Le biais sera alors petit.

On pourrait penser, de manière naïve, qu'une bonne stratégie consisterait à choisir une famille d'hypothèses $\mathcal{H}$ la plus riche possible. Ce serait sans compter sur la deuxième composante importante de l'erreur d'une hypothèse : la *variance*. Celle-ci exprime la part de l'erreur liée aux variations possibles de l'hypothèse h induite à partir de E. En d'autres termes, l'hypothèse h produite par la méthode d'apprentissage n'est pas toujours (et même rarement) l'hypothèse optimale h^*

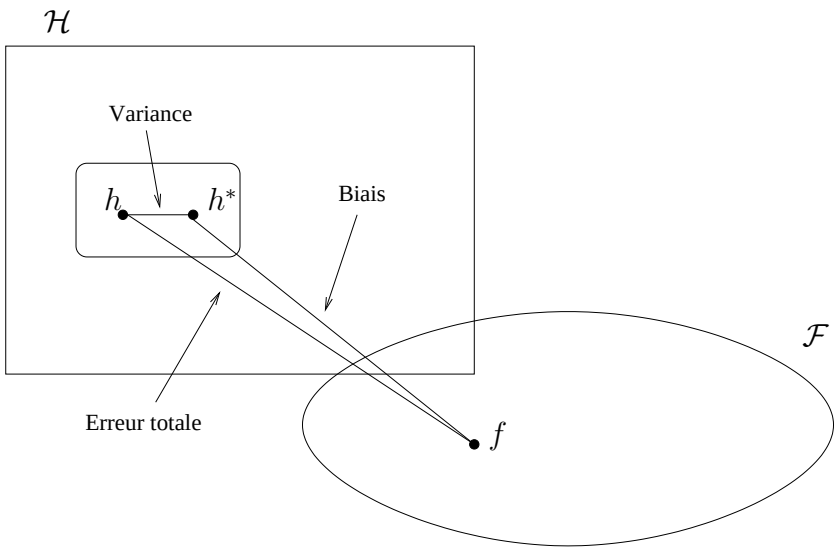


Fig. 1.3 – *La décomposition de l'erreur réelle d'une hypothèse h, en erreur dûe à la variance liée à l'échantillon E, d'une part, et en erreur liée au biais induit par la famille $\mathcal{H}$ d'hypothèses choisie, d'autre part.*

qui s'approcherait le plus de la fonction cible f. La variance exprime alors l'écart entre les deux. Pour une famille d'hypothèses $\mathcal{H}$ peu riche (possédant donc peu de degrés de liberté en termes statistiques), un petit changement dans l'échantillon d'apprentissage E aura peu d'impact sur le choix de l'hypothèse h. Inversement, pour une famille $\mathcal{H}$ très riche, le nombre de paramètres à estimer devient très élevé et la variance sera d'autant plus grande.

En résumé, choisir une bonne famille d'hypothèses nécessite de trouver un compromis idéal entre le biais limitant la richesse de cette famille, et la variance liée à cette famille (Fig. 1.4). Ce compromis est connu dans la communauté sous le terme de *dilemne biais/variance*. En pratique, une variance trop élevée aboutira à la construction d'hypothèses dont le taux d'erreur en apprentissage sera certes relativement faible, mais dont l'erreur en généralisation sera probablement élevée. Une telle situation, dite de *sur-apprentissage*, n'est pas plus souhaitable que celle induite par un biais trop élevé : dans un tel cas, l'obtention d'une hypothèse modélisant de manière satisfaisante le concept cible sera impossible. On parlera alors de *sous-apprentissage*.

Nous pouvons donc constater que la seule observation de l'erreur empirique est insuffisante pour choisir une hypothèse h, et que la prise en compte du biais et de la variance est primordiale. Le cadre d'apprentissage PAC (pour Probablement Approximativement Correct) (Valiant, 1984) nous donne des conditions théoriques pour pouvoir appliquer le principe inductif ERM.

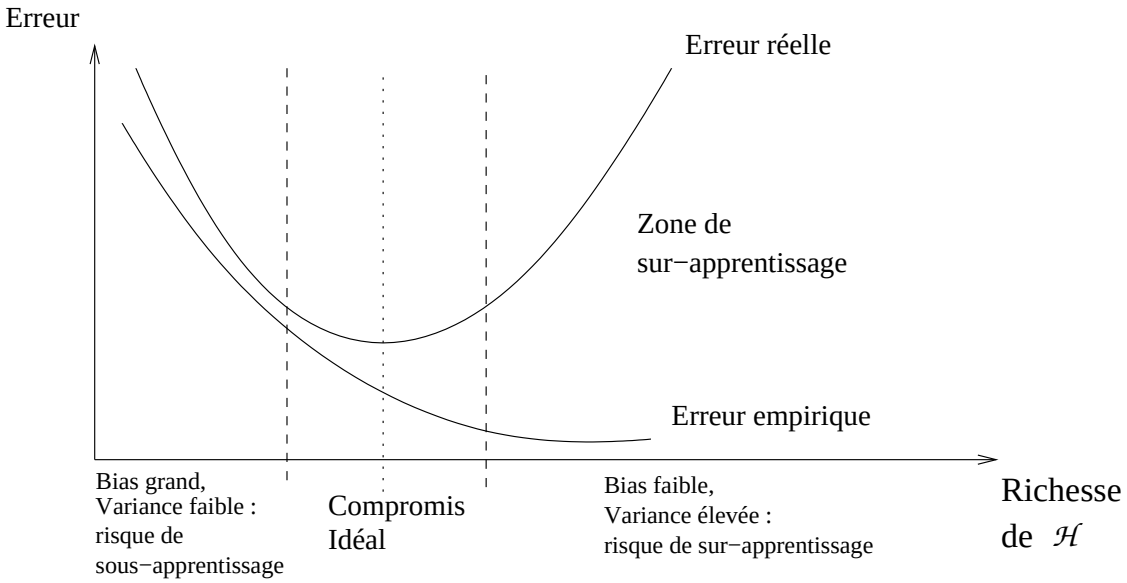


FIG. 1.4 – *Le dilemme biais/variance.*

1.4 Le principe ERM et le cadre PAC

Le cadre d'apprentissage PAC a été proposé par Valiant (1984). Un problème sera dit PAC-apprenable s'il existe une méthode d'apprentissage capable, avec une probabilité arbitrairement élevée, de produire une hypothèse dont les prédictions incorrectes seront arbitrairement peu fréquentes. Plus formellement :

Définition 1.1 *Une classe de problèmes $\mathcal{F}$ sur une population Ω sera dite PAC-apprenable au sens fort s'il existe une méthode d'apprentissage A telle que pour tout problème f de $\mathcal{F}$, pour toute distribution D sur Ω, pour tout $(\delta,\epsilon) \in]0;1/2]^2$, A est capable de fournir en temps polynomial une hypothèse h telle qu'avec une probabilité de $1-\delta$, l'erreur de h soit inférieure à ϵ : $P(err(h) < \epsilon) > 1-\delta$.*

où $err(h)$ désigne la probabilité d'erreur de h sur Ω, c'est à dire

$$err(h) = P_{\omega\in\Omega}(h(x) \neq f(x)). \tag{1.1}$$

Cependant, en pratique, $err(h)$ n'est pas calculable (la taille de Ω pouvant par exemple être infinie). Pour assurer la minimisation de $err(h)$, le principe ERM suggère de minimiser l'erreur empirique $\hat{err}$ obtenue avec h sur l'échantillon d'apprentissage E :

$$\hat{err}(h) \quad = \quad \frac{1}{|E|}|\{e_i \in E : h(x_i) \neq y_i\}| \tag{1.2}$$

$$= \frac{1}{m}\sum_{i=1}^{m} [\![h(x_i) \neq y_i]\!]. \quad (1.3)$$

En fait, le principe d'ERM suggère un lien direct entre l'erreur empirique $e\hat{r}r$ et l'erreur réelle err d'une hypothèse h. Conformément à la loi des grands nombres, et si on assure une distribution uniforme dans Ω des exemples de l'échantillon d'apprentissage, plus leur nombre augmente, et plus l'erreur réelle de la meilleure hypothèse h construite à partir de E a de fortes chances d'être similaire à celle de l'hypothèse optimale h^* pour modéliser f. De manière plus formelle, cela se traduit par l'inégalité suivante :

$$\forall \delta \leq 1, \epsilon \geq 0, \exists\ m,\ P(|err(h_m) - err(h^*)| > \epsilon) < \delta \quad (1.4)$$

où $err(h_m)$ est l'erreur réelle d'une hypothèse h_m retournée par un algorithme d'apprentissage fixé à partir d'un échantillon d'apprentissage E de taille m.

Les conditions permettant d'assurer cette inégalité ont été mises en place dans le cadre du modèle PAC (Valiant, 1984).

Dans le cas d'une famille finie d'hypothèses $\mathcal{H}$, l'inégalité de Hoeffdings, qui est une formulation particulière de la loi des grands nombres, permet de borner l'écart entre les taux d'erreur empirique et réelle d'une hypothèse de $\mathcal{H}$:

$$\forall \epsilon > 0,\ P(\max_{h \in \mathcal{H}} |e\hat{r}r(h_m) - err(h_m)| \geq \epsilon) < 2|\mathcal{H}|e^{-2\epsilon^2 m}. \quad (1.5)$$

Par ailleurs, l'écart entre les erreurs réelles de h_m et de h^* est donné par la borne suivante :

$$\forall \epsilon > 0,\ P(|err(h_m) - err(h^*)| \geq \epsilon) < 2e^{-2\epsilon^2 m}. \quad (1.6)$$

En conséquence des Equations 1.5 et 1.6, la probabilité que la meilleure hypothèse construite à partir de l'échantillon d'apprentissage soit également l'hypothèse optimale sur E augmente avec le nombre d'exemples d'apprentissage. D'autre part, on note que la taille de l'échantillon d'apprentissage devra être d'autant plus élevée que la famille d'hypothèses considérée sera riche.

Dans le cas d'une famille d'hypothèses de cardinal infini, l'inégalité de Hoeffdings n'est plus valable. Les travaux de Vapnik et Chervonenkis (1971) permettent toutefois de préciser une borne dans un tel cas. Celle-ci utilise en fait la dimension de Vapnik-Chervonenkis d'une famille d'hypothèses considérée, qui comme le faisait $|\mathcal{H}|$, va traduire sa richesse, c'est à dire sa capacité à exprimer des concepts complexes. La dimension de Vapnik-Chervonenkis ($VCdim$) correspond à la taille maximale d'un échantillon pour lequel $\mathcal{H}$ arrive à modéliser les concepts correspondant à toutes les partitions (on dit alors que la famille d'hypothèses *pulvérise* l'échantillon). Par exemple, comme le montre la Figure 1.5, la $VCdim$ d'un séparateur linéaire dans un espace à deux dimensions est 3.

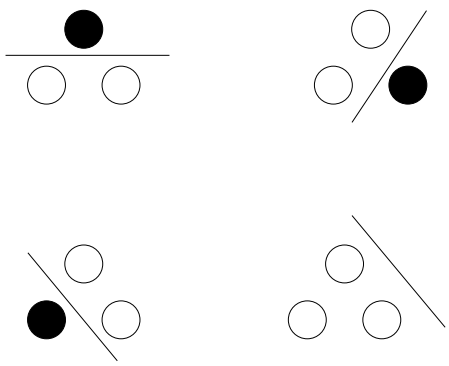


FIG. 1.5 – *Un échantillon de* 3 *exemples pulvérisé par un séparateur linéaire dans le plan (*4 *partitions sur* 8 *sont représentées, les* 4 *restantes étant identiques à une symétrie près). Aucun échantillon de taille* 4 *ne pouvant être pulvérisé par un tel séparateur, la* $VCdim$ *de ce dernier est donc égale à* 3*.*

Les bornes sur $err(h)$ s'expriment alors de la manière suivante :

$$err(h) \quad \leq \quad \hat{err}(h) + O\left(\frac{VCdim(\mathcal{H})}{m}\right). \tag{1.7}$$

La partie droite de l'inégalité 1.7 est composée de deux termes. Le premier traduit la capacité de la méthode d'apprentissage utilisée à apprendre correctement l'échantillon E. On peut donc, de manière intuitive, relier cette quantité au biais de la famille d'hypothèses considérée. Le deuxième terme peut être vu comme un facteur de pénalisation compensant l'éventuel sur-optimisme de $\hat{err}(h)$. Mais si la taille m de E tend à être grande, on aura alors une meilleure confiance en l'erreur empirique. Il est fonction du nombre d'exemples m utilisés pour l'apprentissage, et de la dimension de Vapnik-Chervonenkis de la classe d'hypothèses considérée. Ce deuxième terme peut donc être à rapprocher du concept de variance.

Au delà de la minimisation du risque empirique, cette borne traduit un principe de minimisation du risque *structurel* lié à la sous-famille d'hypothèses utilisée. Intuitivement, elle semble mettre en application le principe du rasoir d'Occam, qui suggère de ne pas *multiplier les entités sans nécessité*. Elle tend en effet à favoriser, parmi plusieurs hypothèses dont les comportements sont similaires sur l'échantillon d'apprentissage, celle qui est générée à l'aide du langage de description ayant la capacité d'expression la plus limitée. Elle suggère également que l'utilisation d'une famille d'hypothèses dont la $VCdim$ est élevée, nécessite un nombre important d'exemples.

En pratique toutefois, il est possible que le nombre d'exemples disponibles ne permette pas à cette inégalité de fournir une borne satisfaisante sur l'écart entre

les erreurs empirique et réelle de l'hypothèse considérée. D'autre part, notons, par exemple, le cas bien connu des séparateurs linéaires dans le plan, nécessitant en théorie plus de 2400 exemples d'apprentissage pour assurer l'inégalité 1.4 avec $\epsilon = 0{,}01$ et $\delta = 0{,}05$ ...

Enfin, une autre difficulté peut être rencontrée. La $VCdim$ n'est en effet pas encore connue formellement pour toutes les familles d'hypothèses $\mathcal{H}$. Le deuxième terme de la partie droite de l'inégalité 1.7 est donc parfois impossible à calculer. Une solution pour contourner le sur-optimisme de $e\hat{r}r(h)$, apporté par une situation de sur-apprentissage, est d'estimer sans biais l'erreur réelle à partir de E. Une stratégie possible pour cette estimation consiste à utiliser le *bootstrap* (Efron & Tibshirani, 1993) : il s'agit de construire l'hypothèse sur un sous-ensemble E' d'exemples tirés aléatoirement avec remise parmi les exemples de E. Une première erreur est alors évaluée sur les exemples de E non présents dans E'. La répétition de ce processus permet d'obtenir une estimation non biaisée de l'erreur réelle. Une autre solution fréquement utilisée consiste à diviser l'échantillon d'apprentissage en deux parties : l'une sera utilisée pour construire l'hypothèse, et l'autre pour estimer l'erreur réelle. Mais la taille de l'échantillon d'apprentissage nécessite de trouver un compromis entre la quantité d'exemples utilisés par la construction d'une hypothèse, et la qualité de son estimation. Les méthodes de *validation croisée* (Efron & Tibshirani, 1995) permettent de contourner ce problème : il s'agit de diviser l'échantillon d'apprentissage en k parties égales, et d'estimer sur la i-ième partie l'erreur réelle de l'hypothèse construite en utilisant les $k-1$ autres parties comme échantillon d'apprentissage. On obtient ainsi k estimations différentes de l'erreur réelle, et chaque exemple de l'échantillon intervient aussi bien dans le processus de construction d'une hypothèse, que dans le processus de validation.

1.5 Intérêt des méthodes ensemblistes

Comme nous l'avons dit, la production d'une hypothèse satisfaisante selon le principe ERM passe par un contrôle du biais et de la variance de la famille d'hypothèses utilisée. Ces deux notions étant antagonistes, il s'agira en fait de trouver un compromis idéal. On peut noter que bon nombre d'algorithmes d'apprentissage tentent de prendre en considération le compromis biais-variance dans le processus d'induction. En d'autres termes, ces approches utilisent des paramètres de généralisation permettant d'éviter des phénomènes de sur-apprentissage. Par exemple, l'algorithme $C4.5$ (Quinlan, 1993) induit des arbres de décision après un processus d'élagage permettant de réduire la taille des arbres (et donc leur $VCdim$) et par conséquent les risques de sur-apprentissage.

Une autre façon d'obtenir un bon compromis entre le biais et la variance consiste à combiner la prédiction d'un sous-ensemble d'hypothèses à l'aide par

exemple d'un vote. Plutôt que d'utiliser une seule hypothèse issue d'une famille $\mathcal{H}$, on considère un ensemble d'hypothèses, chacune d'entre elles ayant été générée soit en utilisant des paramètres de contrôle de biais et de variance différents sur une même famille, soit en utilisant différentes familles d'hypothèses. Ainsi, en combinant les hypothèses, les composantes d'erreurs liées au biais et à la variance tendent à être réduites. En effet :

– Considérons trois hypothèses h_1, h_2 et h_3, issues d'une même famille d'hypothèses. La Figure 1.6 schématise deux situations possibles. Dans le cas où le biais est important (figure de gauche), la famille d'hypothèses utilisée ne permet pas de trouver une modélisation satisfaisante du concept. Combiner plusieurs de ces hypothèses permet de contourner le problème en "émulant" une hypothèse h_{comb} potentiellement extérieure à cette famille. Dans le cas d'une variance élevée (figure de droite), la combinaison d'hypothèses va permettre de réduire cette statistique de dispersion. En effet, une combinaison linéaire d'hypothèses n'étant rien d'autre qu'une moyenne pondérée, il est facile de montrer statistiquement que si σ^2 est la variance d'une hypothèse individuelle de $\mathcal{H}$, la variance d'une combinaison de n hypothèses est égale à $\frac{\sigma^2}{n}$.

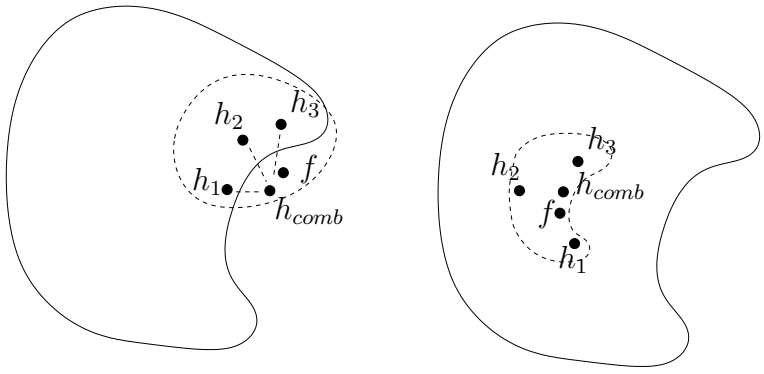


FIG. 1.6 – *L'effet de l'utilisation d'un ensemble d'hypothèses sur le biais (à gauche) et la variance (à droite). La ligne continue symbolise la frontière de la famille d'hypothèses considérée, tandis que la ligne pointillée délimite l'ensemble des hypothèses potentiellement "émulables" par une combinaison de h_1, h_2 et h_3.*

– Si on considère maintenant le cas d'hypothèses provenant de familles différentes, les biais sont intrinsèquement également différents. Il est donc assez immédiat de constater que, les erreurs ne concernant pas les mêmes exemples d'une hypothèse à l'autre, elles auront tendance à s'annuler du fait de la combinaison. Ainsi, il est possible de s'affranchir du biais induit par chacune des familles d'hypothèses considérées, en n'étant pas pénalisé

par l'augmentation de la variance en leur sein, pour les mêmes raisons que celles mentionnées précédemment.

Les méthodes visant à produire un tel ensemble d'hypothèses, et à combiner leurs prédictions, sont dites *ensemblistes*. De telles méthodes ont fait l'objet de nombreux travaux durant la dernière décennie, et forment un domaine à part entière de l'apprentissage automatique.

Par nos remarques concernant l'effet de ces méthodes ensemblistes sur le compromis biais/variance, on constate que les résultats théoriques classiques, issus par exemple du modèle PAC, se retrouvent plus ou moins inappropriés. En effet, alors que l'interprétation en termes de biais et de variance des deux termes de droite de l'inégalité 1.7 suggère un système de vase communiquant entre ces deux termes, il semble qu'il soit possible par combinaison d'hypothèses de réduire les deux, et donc d'avoir des bornes plus petites sur l'erreur réelle. Nous dressons dans le chapitre suivant un état de l'art sur les méthodes ensemblistes, avant de développer l'ensemble des résultats théoriques associés à l'une d'entre elles : le *boosting*.

2 Méthodes Ensemblistes

Résumé : Ce chapitre dresse un état de l'art sur les méthodes ensemblistes, en distinguant deux catégories : les méthodes hétérogènes et les méthodes homogènes. Le boosting, qui fait partie de cette deuxième classe, est présenté en détail : nous faisons un état des lieux des principaux résultats théoriques et extensions.

2.1 Introduction

Les méthodes ensemblistes sont caractérisées par le fait qu'elles combinent, en une hypothèse générale H, un ensemble d'hypothèses dites *de base*, $\{h_1, \ldots, h_n\}$, construites à partir de l'échantillon d'apprentissage E. Deux points clés caractérisent une hypothèse H produite par une méthode ensembliste, et influencent ses performances : premièrement, la façon de générer des hypothèses de base qui la constituent, et deuxièmement, le type de combinaison opéré.

Avant de présenter un état de l'art sur les principales méthodes ensemblistes, notons que de nombreux travaux se sont intéressés aux propriétés qui permettent (sans pour autant garantir) l'obtention d'une hypothèse H performante. Par exemple, Ali et Pazzani (1996), Kuncheva et al. (2002), Kuncheva et Whitaker (2003) étudient, d'un point de vue aussi bien théorique qu'expérimental, le lien entre la *diversité* existant au sein de l'ensemble des hypothèses de base, et les performances de leur combinaison, en exploitant la corrélation entre les erreurs commises par chaque hypothèse h_i, $i = 1 \ldots n$. Dans un contexte différent, les travaux présentés par Lecce et al. (2000) montrent l'intérêt des connaissances préalables (telle que la performance individuelle de chaque hypothèse) dans la combinaison finale.

La présentation de l'état de l'art, telle que nous avons décidé de l'opérer, divise le domaine en deux catégories principales, selon la nature des hypothèses de base,

et donc des méthodes d'apprentissage impliquées dans leur construction :

- **Les méthodes ensemblistes hétérogènes**, qui manipulent un ensemble d'hypothèses de base de natures différentes, c'est à dire produites par différentes méthodes d'apprentissage, à partir d'un unique échantillon d'apprentissage.
- **Les méthodes ensemblistes homogènes**, qui combinent un ensemble d'hypothèses produites par une même méthode d'apprentissage. La diversité sur les hypothèses de base construites s'opère en modifiant la distribution de probabilité de l'échantillon d'apprentissage.

2.2 Méthodes ensemblistes hétérogènes

Ces méthodes permettent la combinaison d'hypothèses produites à partir de méthodes d'apprentissage différentes (donc d'hypothèses de nature hétérogène), appliquées à un même échantillon d'apprentissage. Plus que sur le choix des méthodes d'apprentissage utilisées pour générer les hypothèses, la majorité des travaux du domaine porte sur la manière de les combiner efficacement. Nous dressons dans ce qui suit un éventail des principales techniques qui ont été proposées.

2.2.1 Combinaison par vote

Le vote d'un ensemble d'hypothèses est probablement la méthode la plus simple pour combiner celles-ci. Reprenons la liste des types de vote établie par Bahler et Navarro (2000) :

- Le *vote majoritaire*, au cours duquel chaque hypothèse vote pour l'étiquette qu'elle prédit. La prédiction globale est alors l'étiquette recevant le plus de suffrages.
- Le *vote majoritaire pondéré*, similaire au vote majoritaire simple, à la différence près qu'un poids est associé au vote de chaque hypothèse. L'étiquette prédite par le vote global est alors celle recevant le poids le plus élevé sur l'ensemble des votes. Les poids impliqués reflètent généralement un indice de confiance sur la prédiction de l'étiquette faite par l'hypothèse.
- Le *vote avec seuil*, pour lequel l'étiquette prédite est celle recevant le plus de votes, avec un écart significatif par rapport aux suffrages recueillis par les autres étiquettes. Ce principe peut être également utilisé dans un contexte pondéré.
- L'*unanimité*, pour laquelle une étiquette sera associée à un nouvel individu si toutes les hypothèses de l'ensemble concordent sur celle-ci. Il est donc possible qu'un tel procédé aboutisse à une situation d'indéterminisation

(on parle alors d'abstention). Là encore, il est possible d'adapter ce principe dans un contexte pondéré.

Notons ici que ces combinaisons par vote s'utilisent également pour combiner des hypothèses construites par une seule méthode d'apprentissage à partir d'échantillons différents.

2.2.2 Sélection dynamique d'hypothèses

La sélection dynamique d'hypothèses est comparable au principe de sélection adaptative de la meilleure hypothèse au sein d'une famille. Mais là où certaines méthodes permettent simplement de choisir l'hypothèse jugée la plus pertinente sur l'échantillon d'apprentissage (Tsoumakas et al., 2004), les méthodes de sélection dynamique permettent de choisir, pour chaque nouvel individu, l'hypothèse jugée la plus apte à prédire son étiquette.

Dans ce contexte, Merz (1996) propose par exemple un mécanisme d'évaluation, pour différentes zones géométriques de l'espace de représentation, de la performance de chaque hypothèse de base. La classification d'un nouvel individu est alors effectuée par l'hypothèse jugée la plus pertinente dans la zone où cet individu est situé. Giacinto et Roli (1997) proposent une méthode comparable, où la sélection est basée sur les performances de chaque classifieur sur les N plus proches voisins de l'exemple considéré dans l'échantillon d'apprentissage.

2.2.3 Le Stacking

De manière générale, le terme *stacking* (Wolpert, 1992) fait référence à toute méthode ensembliste consistant à combiner un ensemble d'hypothèses hétérogènes, en utilisant leurs réponses pour un exemple e_i. Ces réponses sont vues comme les attributs d'une nouvelle représentation de e_i, qui est utilisée par une autre méthode d'apprentissage A pour produire une hypothèse finale.

Le principe général est le suivant : il s'agit premièrement de projeter chaque exemple d'apprentissage e_i dans un nouvel espace à K dimensions, à l'aide de K hypothèses de base $h_1, \ldots, h_K$, dites de niveau 0. Le k-ième attribut de ce nouveau vecteur de représentation correspond en fait à l'étiquette $h_k(x_i)$ attribuée par l'hypothèse h_k. On obtient ainsi une nouvelle description $(h_1(x_i), \ldots, h_K(x_i))$ de e_i. Une hypothèse H, dite de niveau 1, est ensuite construite par A sur cette nouvelle représentation des exemples de E.

Attribuer une étiquette à un nouvel individu consiste dans un premier temps à le projeter dans le nouvel espace de représentation en utilisant les hypothèses de niveau 0, puis à présenter le nouveau vecteur ainsi construit à l'hypothèse de niveau 1 pour obtenir l'étiquette.

Il existe néanmoins des approches plus originales du stacking.

Par exemple, Merz (1999) propose une méthode de stacking baptisée SCANN (pour Stacking, Correspondence Analysis, Nearest Neighbors). Les données de niveau 1 y sont formées par les probabilités d'appartenance à chacune des classes attribuées par les hypothèses de niveau 0. Soit N la matrice qui à chaque exemple de l'échantillon associe une ligne, dans laquelle sont stockées ces probabilités. Une méthode d'analyse factorielle des correspondances (AFC) est appliquée sur N, afin d'établir les relations existant entre les lignes et les colonnes de cette matrice. La technique consiste ensuite à exprimer N à l'aide d'un produit de matrices, dont l'une contient une nouvelle représentation des exemples de l'échantillon d'apprentissage, tandis qu'une autre contient la représentation, dans le même espace, des classes du problème. Ainsi, un exemple e_i se verra attribuer la classe c si la représentation x_i de e_i dans ce nouvel espace est proche de celle de c.

De manière plus générale, on note que les travaux récents sur le stacking abordent principalement la question de la construction du nouveau vecteur d'attributs ainsi que celle du choix de la famille d'hypothèses utilisée au niveau 1. L'étude expérimentale présentée par Ting et Witten (1999) aboutit à plusieurs constatations :

- D'une part, et de manière peu surprenante, le stacking se montre plus performant lorsque le nouvel espace de représentation des exemples est constitué, non pas de l'étiquette prédite par les hypothèses de base, mais des probabilités qu'elles associent à chacune des classes.
- Ensuite, les meilleurs résultats obtenus au niveau 1 sont atteints par des méthodes exploitant des modèles de régression linéaire.
- Enfin, et ceci est le point le plus important, le stacking réalise des performances significativement plus élevées qu'un schéma de combinaison par vote classique. A ce sujet, la vaste étude expérimentale effectuée par Dzeroski et Zenko (2002) et exploitant un ensemble d'arbres de décision, montre l'efficacité du stacking non seulement face aux méthodes classiques de vote, mais également par rapport au principe consistant à sélectionner la meilleure hypothèse de l'ensemble.

2.2.4 Les Cascade Generalization

Les méthodes de généralisation en cascade (ou *cascade generalization*) (Gama & Brazdil, 2000) sont inspirées du principe du stacking. En effet, il s'agit là aussi de considérer les prédictions des hypothèses comme de nouveaux attributs de description des individus, et de construire une hypothèse à l'aide de ces nouveaux attributs. Cependant, c'est sur plusieurs niveaux que ce principe est appliqué, et les attributs d'origine sont conservés et complétés itérativement. Le principe est le suivant : une première hypothèse de niveau 0 est construite sur l'ensemble d'attri-

buts initial. On utilise ensuite celle-ci pour étiqueter les exemples de l'échantillon d'apprentissage. On concatène alors l'étiquette prédite au vecteur de représentation de chaque exemple, ce qui revient à dire que l'on ajoute une dimension à l'espace de représentation. Ce nouvel ensemble d'attributs de l'échantillon d'apprentissage est alors présenté à une seconde méthode d'apprentissage différente de la précédente. Il en résulte une nouvelle hypothèse de niveau 1, qui est également utilisée pour étiqueter l'échantillon d'apprentissage. De façon similaire, la réponse de cette hypothèse est alors ajoutée à l'ensemble d'attributs précédent, et ainsi de suite.

Attribuer une étiquette à un nouvel exemple consiste alors à présenter successivement à chaque niveau i sa description courante, puis à compléter celle-ci avec la réponse obtenue par le classifieur du niveau i. L'étiquette de l'individu est ainsi déduite au dernier niveau de la cascade. L'intérêt des méthodes de généralisation en cascade tient dans le fait que les attributs initiaux de description du problème sont enrichis par les étiquettes produites par les hypothèses intermédiaires.

Les méthodes ensemblistes hétérogènes, décrites jusqu'à présent, offrent donc la possibilité de combiner des ensembles d'hypothèses diversifiées, afin de réduire le biais et la variance. Dans la réalité toutefois, il n'est pas toujours "pratique" d'avoir recours à différentes méthodes d'apprentissage. La section suivante passe en revue les méthodes utilisées pour générer des hypothèses diversifiées dans le cas où une seule méthode d'apprentissage est appliquée. La diversité ne s'opère donc plus sur les méthodes d'apprentissage, mais sur l'échantillon d'apprentissage manipulé.

2.3 Méthodes ensemblistes homogènes

Les techniques produisant un ensemble d'hypothèses à partir d'une même méthode d'apprentissage sont généralement appelées méthodes ensemblistes homogènes. Dans ce cas, l'accent est moins mis sur la manière de combiner les modèles que sur la façon de générer une diversité des hypothèses induites. Historiquement, les premières méthodes ensemblistes homogènes ont exploité des réseaux de neurones. Ces méthodes étaient pour la plupart basées sur des choix aléatoires des structures des réseaux, ainsi que des poids synaptiques initiaux (Hansen & Salamon, 1990 ; Opitz & Shavlik, 1996 ; Islam et al., 2003). Depuis, ces techniques se sont largement étendues à d'autres familles d'hypothèses, et comme nous le verrons dans la suite, certaines ont été proposées, générant même une diversité indépendamment de la méthode d'apprentissage utilisée.

2.3.1 Les Random Forests

Nous détaillons ici les méthodes visant à produire des ensembles d'arbres de décision. Ces ensembles sont connus sous le nom de *random forests* (Breiman, 2001). Nous présentons le principe de leur construction.

Rappelons que dans sa forme la plus simple, un arbre de décision partitionne récursivement l'espace de représentation des individus à l'aide d'hyperplans orthogonaux aux axes. Chaque noeud de l'arbre se voit attribuer un test sur la valeur d'un des attributs de représentation, séparant ainsi par un hyperplan les exemples concernés par le test. Le processus continue jusqu'à obtenir des feuilles "pures", c'est à dire ne contenant des exemples que d'une seule classe (voir Fig. 2.1) .

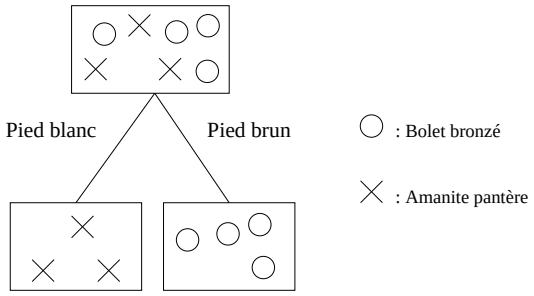


Fig. 2.1 – *Construction d'un noeud de l'arbre de décision sur le problème introductif des champignons : le test de la couleur du pied permet de séparer au mieux les bolets bronzés des amanites pantères.*

Nous avons déjà évoqué en Section 1.4 le fait que la construction d'arbres constitués uniquement de feuilles pures risquait de tendre vers des situations de sur-apprentissage. Si l'élagage apparaît comme une solution efficace, il doit être utilisé avec précaution afin de trouver un compromis satisfaisant entre le biais et la variance. Un élagage trop rigoureux tend en effet à sur-généraliser, et donc à augmenter le biais de l'hypothèse, alors qu'un élagage peu contraignant ne permet plus de généraliser correctement, la variance étant trop forte. Une solution est de construire des ensembles (ou *forêts*) d'arbres (Ho, 1998 ; Dietterich, 2000b ; Breiman, 2001).

Comme dans le cas général des méthodes ensemblistes, les performances d'une forêt d'arbres sont fortement liées à l'indépendance de comportement entre chaque arbre.

Une large étude des forêts d'arbres est présentée par Breiman (2001). Plusieurs méthodes de construction d'ensemble d'arbres y sont introduites, parmi lesquelles

on trouve :

- Une méthode dite de *sélection aléatoire des attributs* qui vise à ne retenir, au cours de la construction d'un arbre, qu'un sous-ensemble d'attributs (tirés aléatoirement) pour la construction d'un noeud. Le test associé à celui-ci sera alors généré en cherchant le meilleur attribut, au sens d'un critère, au sein de cet ensemble.
- Dans le cas où le nombre d'attributs de description du problème est limité, la diversité au sein de la forêt risque d'être faible. Une solution envisagée est basée sur la construction de nouveaux attributs à l'aide de combinaisons linéaires des attributs initiaux : un nouveau noeud se voit attribuer un test choisi parmi un nombre déterminé de combinaison linéaires générées aléatoirement. Quoiqu'efficace, on peut néanmoins constater qu'une telle méthode va tendre à réduire le niveau d'intelligibilité des arbres induits. Il est donc également important de préciser ici qu'en générant une forêt d'arbres, donc en combinant plusieurs hypothèses, l'objectif est clairement plus tourné vers une amélioration des performances de classement que vers un maintien de la compréhensibilité du modèle.

Par ailleurs, Dietterich (2000b) présente une solution consistant à effectuer, pour chaque noeud, un classement des tests les plus efficaces (au sens d'une fonction d'entropie). On sélectionne ensuite aléatoirement, parmi les vingt premiers, le test qui sera attribué au noeud. Les expérimentations présentées montrent que cette méthode appliquée à $C4.5$ produit des forêts significativement plus performantes qu'un arbre seul.

Enfin, la méthode présentée par Ho (1998) construit chaque arbre dans un sous-espace d'attributs sélectionnés aléatoirement. Ce principe porte le nom de *random subspace method*. Les hyperplans séparateurs peuvent être quelconques (*i.e.* ils ne sont pas forcément parallèles aux axes), et les arbres construits ne subissent aucun élagage. Toutefois, la variance élevée est compensée par la combinaison des arbres, celle-ci consistant à attribuer la classe dont la moyenne des probabilités associées par chaque arbre est la plus élevée.

Les méthodes précédemment décrites génèrent une variabilité des hypothèses en modifiant le processus d'induction des arbres. Si nous voulions faire une analogie avec le domaine de la sélection d'attributs (Kohavi, 1994), nous caractériserions de "wrapper" de telles méthodes. Dans les sections suivantes, nous présentons des techniques qui opèrent en amont du processus d'induction. Pour prolonger l'analogie, celles-ci s'apparentent donc plus à des méthodes de type "filter".

classe	code
0	000100
1	100000
2	011010
3	000010
4	110000
5	110010
6	001101
7	001000
8	000100
9	001100

Tab. 2.1 – *Un exemple de codes appliqués à une tâche de reconnaissance de chiffres manuscrits. Chaque bit code une propriété symbolique de ces chiffres. Par exemple, le premier bit est positionné à* 1 *si le caractère de la classe correspondante comporte une ligne verticale (c'est le cas pour les chiffres* 1*,* 4 *et* 5*), et à* 0 *sinon (cf. Table table-sig).*

2.3.2 Méthodes à base de codes correcteurs d'erreurs

Les méthodes d'ECOC (pour Error Correcting Output Codes) permettent de décomposer un problème multiclasse en un ensemble de problèmes binaires, et d'obtenir une hypothèse combinée finale plus performante. Le principe est de partitionner l'ensemble des classes en deux sous-ensembles, qui forment ainsi deux classes d'un nouveau problème d'apprentissage binaire. En construisant plusieurs partitions de ce type, et en construisant une hypothèse pour chacun des problèmes correspondants, le problème est traité par parties.

Dans la pratique, à chaque classe est associée une séquence binaire, appelée *code*, de taille n. Ces codes vont déterminer n partitions possibles de l'ensemble des classes. Leur construction peut être faite soit de manière stochastique, en tirant par exemple chaque bit de façon uniforme dans {0,1}, soit de manière déterministe, en faisant appel aux connaissances d'experts du domaine. Les Tableaux 2.1 et 2.2 présentent un exemple, repris de Dietterich et Bakiri (1995).

Il s'agit ici d'une tâche de reconnaissance de chiffres manuscrits. Chaque bit des codes décrit une propriété d'ordre symbolique des chiffres, telle que la présence de courbes fermées, ouvertes sur la droite, etc. Considérons le i-ième bit de chacun de ces codes ; il détermine une partition de l'ensemble des classes : on regroupe dans un premier sous-ensemble les classes pour lesquelles ce bit a pour valeur 1, et dans un deuxième sous-ensemble, celles pour lesquelles ce bit a pour valeur 0. On obtient ainsi une i-ième partition qui permet de construire une i-ième hypothèse avec un algorithme sachant travailler sur deux classes uniquement.

position	signification
1	Contient une ligne verticale
2	Contient une ligne horizontale
3	Contient une ligne diagonale
4	Contient une courbe close
5	Contient une courbe ouverte à gauche
6	Contient une courbe ouverte à droite

TAB. 2.2 – *Signification de chaque bit des codes pour le problème de reconnaissance de chiffres manuscrits.*

L'étiquetage d'un nouvel individu consiste à construire le code c associé à cet exemple en le présentant à chacune des hypothèses. On associera à cet individu la classe dont le code sera le plus proche de c. La distance de Hamming, qui compte le nombre de bits qui diffèrent entre les deux séquences, est souvent utilisée dans cette tâche.

Berger (1999) présente une application des méthodes d'ECOC à la classification de documents, où les codes sont générés de manière stochastique. Le problème considéré concerne le sujet abordé par un document (par exemple, la politique, l'astronomie, l'espace, etc.). L'hypothèse de base utilisée est un classifieur bayésien naïf, qui suppose que les sujets des documents suivent une distribution de probabilité dictée par les mots qu'ils contiennent. Ainsi, un document W contenant l'ensemble de mots $\{w_1,w_2,\ldots,w_n\}$, aura une probabilité d'appartenance à la classe y donnée par $P(W|y) = p(y).\prod_{i=1}^{N} p(w_i|y)$. Un document se verra attribuer la classe correspondant à la probabilité maximale. Les codes des classes sont choisis de manière aléatoire, et une étude expérimentale (Berger, 1999) montre que le taux d'erreur du modèle global diminue lorsque la taille de ces codes augmente. L'argument avancé en faveur des codes générés de manière aléatoire, est que cette stratégie est plus favorable à une diversité au sein de l'ensemble des hypothèses, que ne peut l'être une génération guidée par les experts du domaine. En effet, cette deuxième solution tend à produire des codes proches pour des classes similaires, par exemple pour les classes "astronomie" et "espace" du problème de classification de documents.

Kong et Dietterich (1995) montrent que les méthodes d'ECOC améliorent les performances d'une méthode d'apprentissage en diminuant aussi bien le biais de la famille d'hypothèses utilisée, que la variance. En effet, le fait de maximiser la distance minimale entre deux codes de classes aboutit à des problèmes décrits par des configurations géométriques très différentes. L'erreur induite par le biais de la famille d'hypothèses considérée est alors très différente d'une hypothèse à l'autre. D'autre part, le problème global étant décomposé en sous-problèmes binaires, les famille d'hypothèses manipulées ont une variance moins élevée.

Les performances du système final sont évidemment fortement dépendantes du choix de l'ensemble des codes. Plus précisément, un tel ensemble devra maximiser la distance de Hamming entre chaque paire de ses éléments. Une telle condition permet par ailleurs d'obtenir un ensemble d'hypothèses commettant des erreurs de manière décorrélée.

2.3.3 Le Bagging

Le bagging (pour *B*ootstrap *agg*regat*ing*) est basé sur un processus stochastique de modification de l'échantillon d'apprentissage E pour créer un ensemble d'hypothèses diversifiées. Cette méthode consiste à construire chaque hypothèse à partir d'un rééchantillonnage par booststrap (tirage aléatoire avec remise) de E. Les hypothèses ainsi construites sont ensuite combinées par un vote. L'algorithme général est donné par le pseudo-code de l'Algorithme 1.

Algorithme 1: Pseudo-code du bagging

Données : Un échantillon d'apprentissage E, un nombre d'itérations T, une méthode d'apprentissage A
Retourner : Une hypothèse globale H_T
pour $t = 1$ **jusqu'à** T **faire**
    E_t =rééchantillonnage(E);
    $h_t = A(E_t)$;
fin
Retourner H_T **tel que**
    $H_T(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right)$

De nombreux travaux ont pu montrer les bienfaits du bagging. Parmi ceux-ci, la vaste étude expérimentale effectuée par Breiman (1996a) présente une nette amélioration des performances de la méthode d'apprentissage A utilisée. Notons que ce gain est conditionné par l'utilisation d'une méthode dite *instable*, c'est à dire pour laquelle une faible perturbation de l'échantillon d'apprentissage entraîne la production d'une hypothèse très différente (nous reviendrons sur cette condition avec le boosting). Par exemple, une méthode de production d'arbres de décision est assez instable. Les études expérimentales mettent en avant des gains de performances considérables lorsque le bagging est appliqué à de telles méthodes. L'explication se trouve dans le fait que la part de l'erreur liée à la variance est élevée pour ces méthodes. Les arbres construits sur les différents échantillons d'apprentissage sont donc très diversifiés. Le bagging améliore donc les performances d'une méthode d'apprentissage en contrôlant la variance au sein

de la famille d'hypothèses utilisée. Au contraire, la méthode des k plus proches voisins (k-PPV) (Cover & Hart, 1967) est une méthode que l'on peut qualifier de stable, et donc moins sujette à des améliorations de performance par bagging. Rappelons que les k-PPV attribuent une étiquette à un nouvel individu en cherchant celle qui est la plus représentée parmi les k exemples les plus proches (au sens d'une distance) dans l'échantillon d'apprentissage. De par ce principe, la perturbation de l'échantillon d'apprentissage n'aura qu'une influence très localisée sur le comportement de l'hypothèse, et le bagging aura donc peu d'effet.

La dernière méthode ensembliste que nous devons aborder dans cet état de l'art est celle du *boosting*. Etant donné qu'il s'agit de la technique à l'origine de l'intégralité des contributions présentées dans cette thèse, nous lui consacrons une section entière.

2.4 Le Boosting

Commençons par un exemple simple désormais bien connu de la communauté du domaine, et proposé par Freund et Schapire (1999) pour illustrer le problème du boosting. Un parieur de courses hippiques, cherchant à maximiser ses gains, décide de créer un programme informatique prédisant le vainqueur d'une course de chevaux. Pour concevoir ce programme, il fait appel à un expert, et lui demande d'expliquer sa stratégie de jeu. Bien évidemment, l'expert a toutes les peines du monde à exprimer "littéralement" des règles automatiques visant à miser sur tel ou tel cheval. Cependant, quand on lui présente une ensemble de courses passées avec leurs résultats, il est tout à fait capable d'extraire quelques règles, du type : "mise sur ce cheval qui a gagné le plus de courses récemment". Si on lui présente une autre série de courses, il sera capable de compléter son expertise par d'autres règles, par exemple, "ce cheval est plus à l'aise sur courtes distances", etc. Bien que chaque règle soit individuellement peu performante, on peut tout au moins espérer qu'elle obtienne des résultats meilleurs qu'un tirage aléatoire. Afin d'obtenir un programme performant, notre parieur doit donc résoudre deux problèmes :

1. Comment doit-il choisir les séries de courses présentées à l'expert, afin que les règles générées soient les plus utiles ?
2. Une fois collectées les différentes règles, individuellement peu performantes, comment combiner celles-ci en une règle de décision unique et efficace ?

Le bagging, présenté plus haut, est une première méthode pour résoudre en partie le problème. Pour répondre à la première question, il applique une stratégie aléatoire. Concernant la deuxième question, il combine les hypothèses générées par un vote non pondéré. Le boosting aborde le problème différemment. La diversité de l'ensemble d'hypothèses est ici générée par un processus déterministe.

Là où les méthodes présentées jusqu'à présent ont fréquemment recours à des techniques stochastiques, le boosting, en réponse à la première question, génère un ensemble diversifié d'hypothèses en modifiant la distribution de l'échantillon en faveur des exemples difficiles à apprendre, c'est à dire les exemples d'apprentissage sur lesquels les hypothèses construites commettent des erreurs. Quant à la seconde question, le boosting combine les hypothèses à l'aide d'un vote, où chaque hypothèse se voit pondérée en fonction de sa pertinence.

2.4.1 Un peu d'histoire

Le principe du boosting a été introduit par Schapire (1990), utilisé à l'origine comme un mécanisme pour montrer l'équivalence entre la PAC-apprenabilité forte et la PAC-apprenabilité faible.

Intéressons-nous quelques instants à ces concepts, en rappelant quelques notions élémentaires de PAC-apprenabilité. Nous nous plaçons dans le cadre de l'apprentissage binaire. Rappelons dans un premier temps la définition de PAC-apprenabilité au sens fort, donnée en Section 1.4 :

Définition 2.1 *Une classe de problèmes $\mathcal{F}$ sur une population Ω sera dite PAC-apprenable au sens fort s'il existe une méthode d'apprentissage A telle que pour tout problème f de $\mathcal{F}$, pour toute distribution D sur Ω, pour tout $(\delta,\epsilon) \in]0;1/2]^2$, A est capable de fournir en temps polynomial une hypothèse h telle qu'avec une probabilité de $1-\delta$, l'erreur de h soit inférieure à ϵ : $P(err(h) < \epsilon) > 1-\delta$.*

Définition 2.2 *Une classe de problèmes $\mathcal{F}$ sur une population Ω sera dite PAC-apprenable au sens faible s'il existe une méthode d'apprentissage A telle que pour tout problème f de $\mathcal{F}$, pour toute distribution D sur Ω, pour tout $\delta \in]0;1]$, il existe $\eta \in]0;1/2]$ telle que A est capable de fournir en temps polynomial une hypothèse h telle qu'avec une probabilité $1-\delta$, l'erreur de h soit inférieure à $\frac{1}{2}-\eta$: $P(err(h) < \frac{1}{2}-\eta) > 1-\delta$.*

Cette définition traduit le fait que la PAC-apprenabilité au sens faible nécessite l'existence d'un algorithme capable de produire une hypothèse plus performante qu'un étiquetage aléatoire.

Afin de démontrer l'équivalence de ces deux définitions, il faut montrer que l'apprenabilité faible implique la PAC-apprenabilité forte (l'implication réciproque étant triviale). Schapire (1990) montre cette implication à l'aide d'un algorithme, et prouve que celui-ci permet de réduire l'erreur d'une hypothèse produite par une méthode d'apprentissage au sens faible.

Le principe de l'algorithme de boosting originel est le suivant :

1. On utilise l'Oracle pour générer un échantillon d'apprentissage E_1 du problème considéré. On utilise une méthode A pour produire une première hypothèse h_1 à l'aide de E_1 (Fig. 2.2).

2. Étant donnée cette première hypothèse, on génère ensuite un deuxième échantillon d'apprentissage E_2, au sein duquel la prédiction de h_1 est équiprobablement correcte ou incorrecte. On utilise à nouveau A pour produire une hypothèse h_2 sur E_2 (Fig. 2.3).
3. On construit finalement un troisième échantillon E_3, constitué d'exemples fournis par l'Oracle, sur lesquels h_1 et h_2 sont en désaccord. On utilise A pour produire une hypothèse h_3 sur E_3 (Fig. 2.4).
4. L'hypothèse finale consiste en un vote majoritaire de h_1, h_2 et h_3.

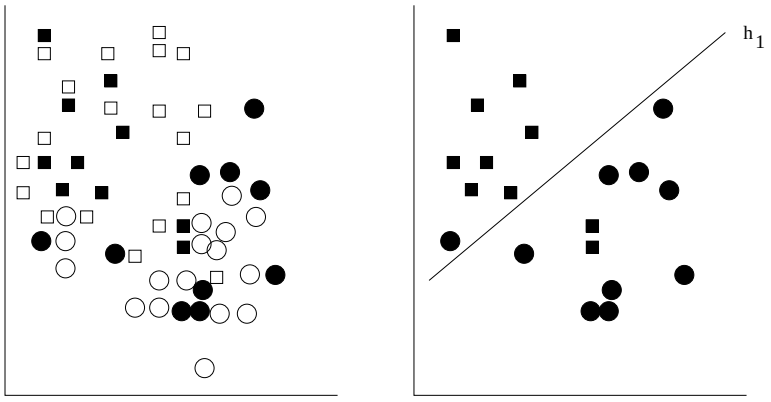


FIG. 2.2 – *A gauche, l'échantillon d'apprentissage E, et la sélection (en noir) des exemples pour construire E_1. A droite, l'hypothèse h_1 générée à partir de E_1.*

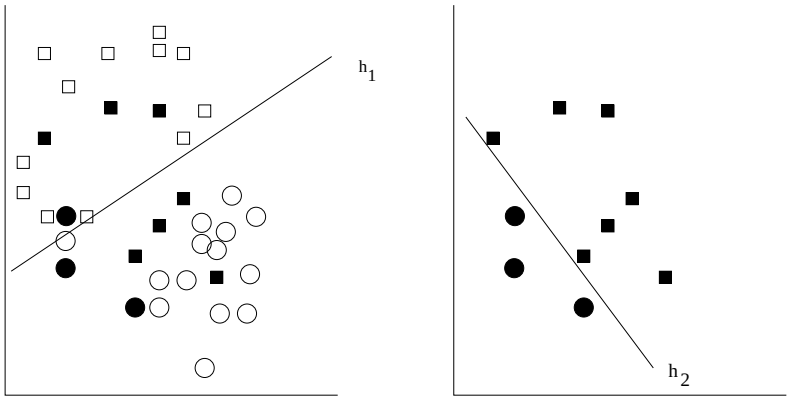


FIG. 2.3 – *A gauche, l'ensemble des exemples de $E \setminus E_1$, et la sélection aboutissant à E_2 d'exemples équiprobablement correctement ou incorrectement étiquetés par h_1 (en noir). A droite, l'hypothèse h_2 construite sur E_2.*

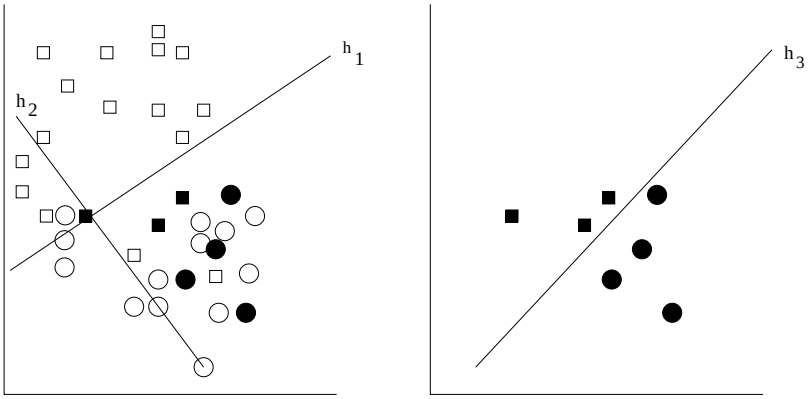


Fig. 2.4 – *A gauche, l'ensemble des exemples de* $(E \setminus E_1) \setminus E_2$*, et la sélection aboutissant à* E_3 *des exemples sur lesquels* h_1 *et* h_2 *sont en désaccord. A droite, l'hypothèse* h_3 *construite sur* E_3*.*

Comme le montre la Figure 2.5, cette procédure permet d'obtenir une hypothèse globale dont le taux d'étiquetage incorrect dans l'échantillon d'apprentissage est réduit. De plus, la répétition de ce processus permet d'obtenir des hypothèses dont la combinaison réduit toujours plus l'erreur. Autrement dit, on peut PAC-apprendre au sens fort en combinant des hypothèses qui ont été PAC-apprises au sens faible. Schapire a ainsi pu donner la preuve (formelle) que les deux notions d'apprentissage étaient équivalentes. Par la suite, Freund et Schapire ont fait évoluer sur cet algorithme pour le rendre opérationnel, mais il est intéressant de noter que les clefs de la preuve se retrouvent encore dans les développements théoriques actuels sur le boosting.

2.4.2 Approche standard du boosting

La véritable émergence du boosting a coïncidé avec la sortie de AdaBoost (pour *Ada*ptative *Boost*ing) (Freund & Schapire, 1996a), dont les détails sont donnés par le pseudo-code de l'Algorithme 2. Comme nous le verrons dans la suite, la quasi-totalité des travaux visant à améliorer le boosting (gestion du bruit, amélioration de la convergence, parallélisation, etc.) ont pris cet algorithme comme base, durant la dernière décennie.

Son principe général est le suivant : il s'agit d'apprendre T hypothèses (dites *faibles*) à l'aide d'une méthode d'apprentissage WL sur T distributions de probabilités $\mathbf{w}_t$ d'un échantillon d'apprentissage E, puis à combiner ces hypothèses faibles h_t en une seule hypothèse H_T "forte". A chaque itération, la mise à jour de la distribution $\mathbf{w}_t$, point crucial d'AdaBoost, favorise (exponentiellement) les exemples mal appris par l'algorithme à l'étape précédente. Détaillons plus

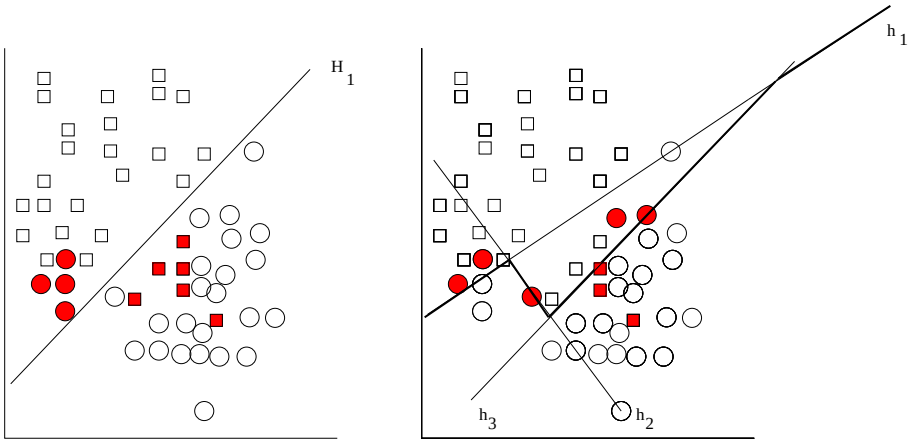


FIG. 2.5 – *A gauche, une hypothèse H_1 produite par l'apprenant faible sur E. A droite, la combinaison H de h_1, h_2 et h_3. Alors que H_1 commet 10 erreurs (en noir), H n'étiquette incorrectement que 8 exemples.*

précisément ce principe : on dispose d'un ensemble d'apprentissage E constitué d'exemples $e_i = (x_i, y_i)$ où $y_i \in \{-1, +1\}$ est l'étiquette de x_i, lui-même étant un vecteur de représentation à p dimensions. On définit une première distribution $\mathbf{w}_1$ uniforme sur E. Puis, à chaque itération t, on utilise WL sur E et $\mathbf{w}_t$ pour obtenir une hypothèse faible h_t. On repondère ensuite les exemples d'apprentissage en construisant une distribution $\mathbf{w}_{t+1}$ à partir de $\mathbf{w}_t$: on augmente exponentiellement le poids des exemples mal classés par h_t (c'est-à-dire ceux pour lesquels $y_i h_t(x_i) = -1$), et on diminue exponentiellement le poids des exemples bien classés par h_t (c'est-à-dire ceux pour lesquels $y_i h_t(x_i) = +1$) ; cette repondération est fonction d'un coefficient c_t dépendant de l'erreur en apprentissage $e\hat{r}r(h_t)$ de h_t. Ce processus est répété T fois. L'hypothèse finale H_T est la combinaison des T hypothèses faibles ainsi construites, pondérées par les coefficients c_t.

Les premières études expérimentales portant sur ADABOOST, présentées par Freund et Schapire (1996a), aboutissent à deux constatations. La première n'est pas surprenante : puisque l'algorithme concentre ses efforts sur les exemples mal classés, l'erreur empirique décroît au fur et à mesure des itérations. La deuxième est quant à elle assez contre intuitive : l'erreur en généralisation (estimée par exemple à l'aide d'une validation croisée) diminue également, et ce même lorsque l'erreur en apprentissage a convergé vers zéro. Ceci semble clairement aller à l'encontre des principes théoriques vus précédemment. En effet, le nombre croissant d'hypothèses de base qui composent H_T augmente en théorie sa complexité, et donc la $VCdim$. Il semblerait logique alors de voir apparaître un phénomène de sur-apprentissage de plus en plus prononcé au fil des itérations du boosting,

Algorithme 2: Pseudo-code d'ADABOOST.

Input : Un échantillon d'apprentissage E, un nombre d'itérations T, un apprenant faible WL
Output : Une hypothèse globale H_T
pour $i = 1$ **jusqu'à** m **faire**
  $w_1(e_i) = 1/m$;
fin
pour $t = 1$ **jusqu'à** T **faire**
  $h_t = \text{WL}(E, \mathbf{w}_t)$;
  $\hat{err}(h_t) = \sum_{e_i \ t.q. \ y_i \neq h_t(x_i)} w_t(e_i)$;
  $c_t = (1/2)\ln((1 - \hat{err}(h_t))/(\hat{err}(h_t))$;
  $Z_t = \sum_{i=1}^{m} w_t(e_i) \exp(-c_t y_i h_t(x_i))$;
  /* Z_t est un coefficient de normalisation*/
  pour $i = 1$ **jusqu'à** m **faire**
    $w_{t+1}(e_i) = w_t(e_i) \exp(-c_t y_i h_t(x_i)) / Z_t$;
  fin
fin
$f_T(x) = \sum_{t=1}^{T} c_t h_t(x)$;
retourner H_T **tel que**
  $H_T(x) = \text{sign}(f_T(x))$

d'autant plus que le principe de repondération des exemples mal appris tend à favoriser incrémentalement un groupe restreint d'exemples, accentuant ainsi le phénomène de sur-spécialisation des hypothèses sur ces exemples. Les résultats expérimentaux ont donc contredit ces *a priori* (voir Fig. 2.6). Par ce phénomène surprenant, ce sont donc les bases mêmes des principales bornes en théorie de l'apprentissage qui ont été remises en cause. Ces bornes sont donc insuffisantes pour modéliser le comportement d'un algorithme "boosté".

Nous présentons dans les deux paragraphes suivants les principaux résultats théoriques sur ADABOOST, justifiant ses performances. Ces résultats concernent principalement des théorèmes fournissant des bornes et des vitesses de convergence sur l'erreur empirique et l'erreur réelle. Notons que nous ne donnerons ici que les théorèmes ainsi que leurs interprétations, les preuves (parfois longues, mais simples) se trouvant dans les trois articles de référence du boosting (Freund & Schapire, 1997 ; Schapire et al., 1997 ; Schapire & Singer, 1999).

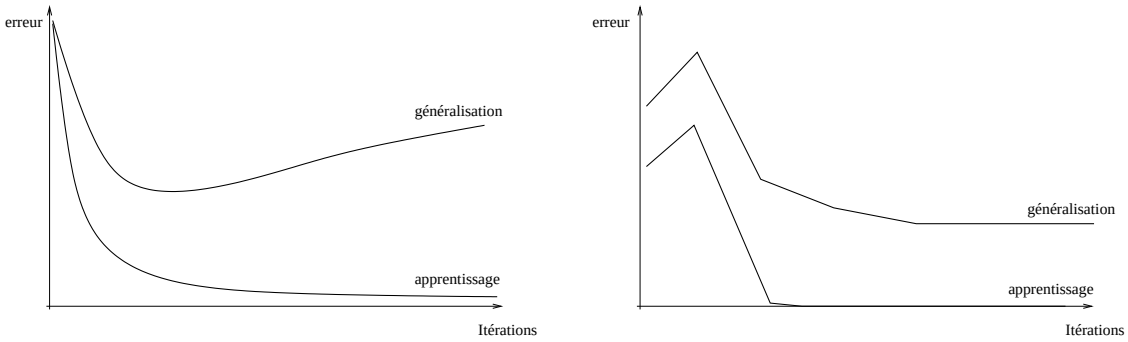


FIG. 2.6 – *A gauche, le comportement "attendu", selon les résultats classiques en Apprentissage, du boosting en termes d'erreurs empirique et réelle : alors que l'erreur en apprentissage diminue, l'erreur en généralisation devrait augmenter avec un nombre important d'itérations. A droite, la tendance réelle obtenue avec* ADABOOST *: alors que l'erreur en apprentissage a convergé vers* 0*, l'erreur en généralisation continue à décroître avant de converger vers un minimum.*

2.4.3 Résultats sur l'erreur empirique

La première propriété d'ADABOOST est sa capacité à diminuer l'erreur en apprentissage de la combinaison H_T des apprenants faibles. Schapire et Singer (1999) bornent celle-ci de la manière suivante :

Théorème 2.1

$$\hat{err}(H_t) = \frac{1}{m}|\{e_i \in E : H_T(x_i) \neq y_i\}| \leq \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t=1}^{T} Z_t.$$

La conséquence de ce résultat est que la minimisation du taux d'erreur en apprentissage nécessite de minimiser, à chaque itération t, le coefficient de normalisation Z_t d'ADABOOST. Intuitivement, ce résultat paraît logique puisque Z_t n'est rien d'autre que la somme des poids mis à jour et que la composante principale de cette somme est portée par les poids des exemples mal classés : ils sont augmentés exponentiellement à chaque itération.

Nous allons voir maintenant que la minimisation de Z_t permet de déterminer le coefficient c_t :

Théorème 2.2 *La minimisation de Z_t est assurée en fixant c_t comme suit :*

$$c_t = \frac{1}{2}\ln(\frac{1 - \hat{err}(h_t)}{\hat{err}(h_t)}).$$

Comme nous serons amenés à réutiliser dans les chapitres suivants le principe de la preuve de ce théorème, nous en donnons ici le détail. Posons tout d'abord :

$$\begin{aligned} W_t^+ &= \sum_{\{e_i=(x_i,y_i)\in E:\ y_i h_t(x_i)=+1\}} w_t(e_i), \\ W_t^- &= \sum_{\{e_i=(x_i,y_i)\in E:\ y_i h_t(x_i)=-1\}} w_t(e_i). \end{aligned}$$

En d'autres termes, W_t^+ est la somme des poids des exemples bien classés par l'hypothèse courante, tandis que W_t^- est la somme des poids des exemples mal classés par celle-ci. Clairement, on a :

$$\begin{aligned} W_t^- &= \hat{err}(h_t), \\ W_t^+ &= 1 - \hat{err}(h_t). \end{aligned}$$

Étudions maintenant la valeur de Z_t en fonction de W_t^+ et W_t^- :

$$\begin{aligned} Z_t &= \sum_i w_t(e_i) \exp(-c_t y_i h_t(x_i)) \\ &= \sum_{\{e_i=(x_i,y_i)\in E:\ y_i h_t(x_i)=-1\}} w_t(e_i)\exp(c_t) + \sum_{\{e_i=(x_i,y_i)\in E:\ y_i h_t(x_i)=+1\}} w_t(e_i)\exp(-c_t) \\ &= W_t^- \exp(c_t) + W_t^+ \exp(-c_t) \end{aligned}$$

On cherche à minimiser Z_t en fonction de c_t, donc :

$$\begin{aligned} \frac{\partial Z_t}{\partial c_t} &= 0 \\ W_t^- \exp(c_t) - W_t^+ \exp(-c_t) &= 0 \\ W_t^- \exp(c_t) &= W_t^+ \exp(-c_t) \\ \exp(2c_t) &= \frac{W_t^+}{W_t^-} \\ c_t &= \frac{1}{2}\ln(\frac{W_t^+}{W_t^-}). \end{aligned}$$

Par conséquent, d'après les définitions de W_t^+ et W_t^- :

$$c_t = \frac{1}{2}\ln\left(\frac{W_t^+}{W_t^-}\right) = \frac{1}{2}\ln\left(\frac{1-\hat{err}(h_t)}{\hat{err}(h_t)}\right).$$

De plus, en remplaçant c_t par sa valeur dans Z_t, on obtient :

$$Z_t = 2\sqrt{W_t^-.W_t^+} = 2\sqrt{\hat{err}(h_t).(1-\hat{err}(h_t))}.$$

Freund et Schapire (1997) montrent ensuite que l'erreur empirique peut décroître de manière exponentielle avec le nombre T d'itérations du boosting. Cependant, ceci n'est vrai que lorsqu'on travaille avec des *hypothèses faibles*, c'est à dire des hypothèses faisant mieux que l'aléatoire. Pour définir ceci formellement, on pose :

$$\gamma_t = \sum_{\{e_i=(x_i,y_i)\in E\}} w_t(e_i).y_i.h_t(x_i).$$

γ_t, qu'on appelle le *seuil* de h_t (*edge* en anglais (Meir & Raetsch, 2003)) est l'espérance de la correction de l'hypothèse h_t ; plus γ_t est proche de $+1$, plus h_t donne de réponses correctes sur l'échantillon d'apprentissage. A l'inverse, plus γ_t est proche de -1, plus h_t classe incorrectement les données d'apprentissage. Supposer que h_t est une hypothèse faible revient donc à supposer $\gamma_t > 0$.

En outre, compte tenu de la définition de γ_t, on a :

$$\gamma_t = W_t^- - W_t^+,$$

et comme $W_t^- + W_t^+ = 1$, on en déduit :

$$\begin{aligned} W_t^+ &= (1+\gamma_t)/2, \\ W_t^- &= (1-\gamma_t)/2, \\ c_t &= \frac{1}{2}\ln\left(\frac{1-\gamma_t}{1+\gamma_t}\right), \\ Z_t &= \sqrt{1-\gamma_t^2}. \end{aligned}$$

Supposer que h_t est une hypothèse faible ($\gamma_t > 0$) a donc au moins deux conséquences. D'une part, tous les coefficients c_t sont strictement positifs. D'autre part, comme

$$\hat{err}(h_t) = W_t^- = \frac{1}{2} - \frac{\gamma_t}{2},$$

il est clair que h_t est une hypothèse faible au sens de la PAC-*apprenabilité faible* (voir Définition 2.2).

Pour finir, on peut montrer :

Théorème 2.3

$$\prod_t Z_t = \prod_t \sqrt{1-\gamma_t^2} < \exp\left(-\sum_t \gamma_t^2\right).$$

Ce théorème signifie que l'on peut exprimer Z_t en une fonction exponentielle des γ_t. Par conséquent, l'erreur en apprentissage baisse de manière exponentielle dès lors que l'on s'assure que chaque hypothèse h_t fait mieux qu'un étiquetage aléatoire.

2.4.4 Résultats sur l'erreur réelle

Analysons à présent le comportement de ADABOOST en généralisation. Une première borne sur l'écart entre l'erreur empirique et l'erreur réelle est donnée par Freund et Schapire (1997) comme suit :

$$err(H_t) < e\hat{r}r(H_t) + O\left(\sqrt{\frac{T.VCdim(\mathcal{H})}{m}}\right)$$

où $\mathcal{H}$ désigne la famille d'hypothèses faibles utilisée. Cette première borne semble donc indiquer une tendance au sur-apprentissage lorsque le nombre d'hypothèses augmente. En pratique toutefois, comme nous l'avons déjà dit, un tel phénomène ne se produit pas, et l'erreur en généralisation continue de diminuer même lorsque l'erreur empirique devient nulle. Cette borne n'est donc pas suffisante pour décrire le comportement d'ADABOOST en généralisation.

Une borne plus "convaincante" a été formulée par Schapire et al. (1997). Celle-ci utilise notamment la notion de *marge* d'un exemple. La marge traduit l'écart à la frontière de décision d'un exemple. Dans le cas du boosting, elle se définit ainsi :

$$m_f(x_i,y_i) = \frac{y_i f_T(x_i)}{\sum_t c_t} = \frac{y_i \sum_t c_t h_t(x_i)}{\sum_t c_t}$$

La marge est donc comprise dans l'intervalle $[-1;1]$, et traduit la confiance dans la prédiction de f_T. Schapire et al. (1997) montrent qu'ADABOOST minimise exponentiellement la borne supérieure sur les marges de l'échantillon d'apprentissage. Ce résultat est apporté par le théorème suivant :

Théorème 2.4 *Pour tout θ,*

$$P_E(y_i f_T(x_i) \leq \theta) \leq 2^T \prod_{t=1}^{T} \sqrt{e\hat{r}r(h_t)^{1-\theta}.(1-e\hat{r}r(h_t))^{1+\theta}}$$

En posant γ tel que $e\hat{r}r(h_t) < 1/2 - \gamma$ pour $1 < t < T$, Schapire et al. (1997) montrent que si $\theta < \gamma$, alors $P_E(y_i f(x_i) \leq \theta)$ décroit exponentiellement vite en fonction de T. Autrement dit, la proportion des exemples d'apprentissage avec une marge faible diminue exponentiellement, *i.e.*, celle des exemples avec une forte marge augmente.

Pour finir, ils démontrent la borne sur l'erreur réelle suivante :

Théorème 2.5

$$err(H_t) \leq P_E(m_{f_T}(x,y) \leq \theta) + O\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

Notons que le terme de droite de cette borne ne dépend pas du nombre T d'itérations. Mais le fait que la probabilité d'une marge faible décroisse au fur et à mesure des itérations permet d'expliquer, en partie, la décroissance de l'erreur en généralisation. Ce résultat, qui est loin d'être satisfaisant, a fait l'objet de nombreuses autres études, par exemple (Rätsch & Warmuth, 2002 ; Koltchinskii & Panchenko, 2002).

2.5 Extensions du boosting

Les travaux présentés dans cette section donnent un certain nombre de pistes sur les travaux qui étendent le cadre standard du boosting. Nous les présentons à titre indicatif.

2.5.1 Le boosting comme méthode de descente de gradient

Les techniques de descente de gradient sont utilisées pour trouver le minimum d'une fonction. Il s'agit d'évoluer par pas successifs sur la courbe de cette fonction en suivant la pente la plus forte, et avec une longueur de pas donnée. Le pas t calcule la valeur courante x_t comme suit :

$$x_t = x_{t-1} - \epsilon \nabla F(x_{t-1})$$

où ϵ caractérise la longueur du pas, et $\nabla F(x_{t-1})$ est le gradient de F en x_{t-1}, qui indique les variations de F autour de x_{t-1}. Une telle méthode se révèle intéressante dans le cadre des méthodes ensemblistes. La fonction à minimiser est alors une fonction de coût $C : lin(\mathcal{H}) \mapsto \mathbb{R}$, avec $lin(\mathcal{H})$ désignant l'espace des combinaisons linéaires d'hypothèses de $\mathcal{H}$. Étant donné $F \in lin(\mathcal{H})$, un pas de descente consiste à trouver $f \in \mathcal{H}$ tel que $C(F + \epsilon f) < C(F)$. Ainsi, chaque itération consiste en la construction de l'hypothèse $f = -\nabla C(F)$. En pratique toutefois, de par la nature de $\mathcal{H}$, il ne sera pas possible de trouver exactement l'hypothèse satisfaisante. On peut toutefois l'approximer, en remarquant qu'on peut approcher $C(F - \epsilon f)$ par $C(F) - \epsilon f^2$. Ainsi, si $\langle , \rangle$ définit un produit interne sur $lin(\mathcal{H})$, l'hypothèse f adéquate sera celle qui maximise $-\langle \nabla C(F), f \rangle$.

Plus précisément, soit $\{(x_1,y_1),\dots(x_m,y_m)\}$ l'échantillon d'apprentissage. Définissons le produit interne suivant :

$$\langle F,G\rangle = \frac{1}{m}\sum_{i=1}^{m} F(x_i)G(x_i),$$

ainsi que la fonction de coût :

$$C(F) = \frac{1}{m}c(y_iF(x_i))$$

où c est une fonction de coût dérivable. Ici, $y_iF(x_i)$ traduit une valeur de marge associée à l'exemple x_i par la fonction F. Alors :

$$-\langle \nabla C(F),f\rangle = \frac{1}{m^2}y_if(x_i)c'(y_iF(x_i)) \tag{2.1}$$

Une fonction de coût sur les marges étant monotone décroissante, $-c'()$ est toujours positif. Ainsi, trouver l'hypothèse f maximisant 2.1 revient à trouver un f minimisant l'erreur sur la distribution définie par :

$$D(x_i) = \frac{c'(y_iF(x_i))}{\sum_{j=1}^{m} c'(y_jF(x_j))}$$

En choisissant la fonction exponentielle comme fonction de coût, le fonctionnement de cette méthode est le même que celui de AdaBoost.

2.5.2 Le boosting et la théorie des jeux

La théorie des jeux offre un cadre dans lequel le boosting peut trouver une interprétation. Considérons un jeu de somme nulle[1], à deux joueurs. Le jeu peut être modélisé par une matrice M. L'un des deux joueurs choisit une ligne, tandis que le second choisit simultanément une colonne. $M(i,j)$ définit la perte du premier joueur (resp. le gain du second) lorsque celui-ci choisit la ligne i (resp. la colonne j). Ainsi, si l'on définit, à l'aide de vecteurs P et Q, une probabilité sur les choix des lignes et des colonnes par les joueurs, l'espérance du coût est donnée par P^TMQ. Il est possible de transposer le boosting dans ce paradigme.

Considérons la famille d'hypothèses $\mathcal{H} = \{h_1,\dots,h_n\}$, et l'échantillon d'apprentissage $(x_1,y_1),\dots,(x_m,y_m)$. La matrice M est alors définie comme suit :

$$M = \begin{cases} 1 & si \quad h_j(x_i) = y_j, \\ 0 & sinon, \end{cases}$$

[1] Un jeu de somme nulle est un jeu où la somme des gains de tous les joueurs est égale à 0, comme le tarot, par exemple.

en considérant que l'algorithme de boosting est le joueur associé aux lignes, et l'apprenant faible est le joueur associé aux colonnes.

Le théorème du minimax, établi par Von Neumann, stipule que :

$$\max_Q \min_P P^T M Q = \min_P \max_Q P^T M Q.$$

Dans le cadre du boosting, on peut utiliser ce théorème en supposant que pour toute distribution sur l'échantillon d'apprentissage, il existe une hypothèse dont l'erreur est inférieure à $1/2 - \gamma$. Dans ce cas,

$$\min_P \max_h M(P,h) \leq \frac{1}{2} - \gamma.$$

Le théorème du minimax indique alors que

$$\max_Q \min_i M(i,Q) \leq \frac{1}{2} - \gamma \leq \frac{1}{2}.$$

En d'autres termes, il existe donc une majorité d'hypothèses qui étiquettent correctement tous les exemples de l'échantillon d'apprentissage.

AdaBoost peut donc être vu comme le cas particulier d'un algorithme plus général de résolution d'une matrice de jeu. En outre, la solution d'un tel problème peut être obtenue par programmation linéaire, et il est donc possible d'établir un lien entre la programmation linéaire et le boosting (Rätsch et al., 2000).

2.5.3 Le boosting et les problèmes multiclasses

L'utilisation d'une méthode de boosting telle que AdaBoost est délicate dans le cas d'un problème multiclasse. En effet, les résultats théoriques sur l'erreur en apprentissage sont conditionnés par les performances de la méthode d'apprentissage utilisée dans le cas binaire : les hypothèses produites doivent avoir un taux d'erreur en apprentissage inférieur à $1/2$, c'est à dire un peu meilleur que l'aléatoire. Il n'en va pas de même lorsque le problème comporte $k > 2$ classes. Il peut d'avérer alors plus difficile d'obtenir un tel taux de succès, dans la mesure où une méthode légèrement plus performante qu'un étiquetage aléatoire ne garantira qu'une performance supérieure à $1/k$. S'il est toutefois possible d'obtenir des hypothèses suffisamment performantes pour garantir la convergence du taux d'erreur, la construction d'hypothèses simples est avantageuse en termes de coût computationnel.

Une première solution présentée dans (Freund & Schapire, 1997) consiste à utiliser comme apprenants faibles des hypothèses associant plusieurs étiquettes à un exemple. En construisant itérativement de telles hypothèses, on obtient une hypothèse globale attribuant à un nouvel individu l'étiquette la plus fréquemment

donnée par les hypothèses faibles. Cette stratégie nécessite l'utilisation d'une pseudo-fonction de coût qui pénalise l'absence de l'étiquette correcte, ainsi que la présence d'étiquettes incorrectes, fournies par h_t. La distribution sur l'échantillon d'apprentissage est une fonction $\tilde{D} : \{x_1, \ldots, x_m\} \times Y \mapsto [0,1]$, où Y désigne l'ensemble des étiquettes du problème. Cette distribution concentre l'apprenant faible sur les paires $\langle x_i, y_k \rangle$, pour $i \neq k$. Ainsi, à l'itération t, l'hypothèse faible minimisera le coût :

$$\tilde{\epsilon} = \frac{1}{2} \sum_{i=1}^{m} \sum_{l \in Y} \tilde{D}_t(x_i,l).([\![y_i \notin \tilde{h}_t(x_i)]\!] + [\![y_i \notin \tilde{h}_t(x_i)]\!])$$

où $[\![p]\!]$ vaut 1 si p est vrai, et 0 sinon. Les auteurs montrent que l'erreur en apprentissage est bornée par un terme qui décroit exponentiellement vite, en fonction de $\sum_{t=1}^{T} \tilde{\gamma}_t^2$, avec $\tilde{\epsilon}_t = 1/2 - \tilde{\gamma}_t$. Cependant, cette méthode nécessite l'utilisation d'une méthode d'apprentissage capable de manipuler des distributions telles que $\tilde{D}_t$.

Une autre solution, présentée dans (Schapire, 1997), combine le principe des codes correcteurs d'erreur (présentés en Section 2.3.2) à la méthode précédente. A chaque itération t, l'apprentissage est focalisé sur les exemples difficiles, mais un *coloriage* différent du problème est considéré. Soit Y l'ensemble des classes du problème. Un coloriage est une fonction $\mu_t : Y \mapsto \{0,1\}$, qui partitionne Y en deux parties. Celle-ci induit un nouveau problème sur l'échantillon d'apprentissage, sur lequel la méthode d'apprentissage produit une hypothèse h_t. En réalité, on peut associer à h_t une fonction similaire à celles qui ont été introduites précédemment :

$$\tilde{h}_t = \mu_t^{-1}(h_t(x)) = \{l \in Y | h_t(x) = \mu_t(l)\},$$

ce qui revient à considérer qu'une hypothèse construite sur un coloriage μ_t prédit pour x_i l'ensemble des étiquettes dont le coloriage est le même que y_i. Celle-ci permet de maintenir une distribution $\tilde{D}$ comme définie précédemment, et grâce à laquelle peut être construite une distribution D_t :

$$D_t(x_i) = \frac{\sum_{l \in Y} \tilde{D}_t(x_i,l) E_t(x_i,l)}{\sum_{i=1}^{m} \sum_{l \in Y} \tilde{D}_t(x_i,l) E_t(x_i,l)},$$

où $E(x_i,l) = [\![\mu_t(x_i) \neq \mu_t(l)]\!]$. Cette distribution tend à concentrer l'apprenant faible, pour un exemple x_i, sur les étiquettes fréquement prédites, et dont le coloriage diffère de y_i. La fonction de coût est alors donnée par :

$$\tilde{\epsilon}_t = \frac{1}{2} \sum_{i=1}^{m} \sum_{l \in Y} \tilde{D}_t(x_i,l)(1 - E_t(x_i,l)) + \sum_{i=1}^{m} \eta_t(i) \sum_{l \in Y} \tilde{D}_t(x_i,l) E_t(x_i,l)$$

où $\eta_t(x_i) = [\![h_t(x_i) \neq \mu_t(y_i)]\!]$. Cette fonction pénalise la prédiction d'un coloriage ne contenant pas y_i, et dans une moindre mesure la présence de représentants d'étiquettes différentes de y_i dans le coloriage prédit.

Ainsi, le coloriage doit être choisi à chaque itération de manière à maximiser :

$$U_t = \sum_{i=1}^{m} \sum_{l \in Y} \tilde{D}_t(x_i,l)[\![\mu_t(l) \neq \mu_t(y_i)]\!].$$

Plusieurs principes de construction de coloriage sont présentés dans (Schapire, 1997), parmi lesquels un choix aléatoire.

Les bornes de convergence de l'erreur empirique vers l'erreur réelle sont ici déduites de celles présentées dans (Freund & Schapire, 1997).

Nous avons évoqué dans cette section quelques axes de recherche concernant le boosting. Il reste cependant à aborder une catégorie particulière de ces travaux, celle qui concerne l'utilisation de données d'apprentissage bruitées.

La présence de bruit sur les étiquettes, c'est à dire d'exemples d'apprentissage dont l'étiquette est erronée, affecte en effet particulièrement AdaBoost, et a des conséquences néfastes sur les performances de l'hypothèse globale en termes d'erreur réelle.

Dans la partie suivante, nous donnons de plus amples détails sur ce problème, puis nous présentons la première contribution mise en place dans le cadre de cette thèse.

Deuxième partie

Adaptation du boosting aux données bruitées

Les travaux présentés dans cette partie ont donné lieu aux deux publications internationales suivantes :

- Janodet, J.C., Nock, R., Sebban, M. & Suchier, H.M. (2004). Boosting grammatical inference with confidence oracles. *International Conference on Machine Learning (ICML'04)* (pp. 425–432).
- Sebban, M., & Suchier, H.M. (2003). On boosting improvement: Error reduction and convergence speed-up. *14th European Conference on Machine Learning (ECML'03)* (pp. 349–360).

A la suite des conférences nationales *CAp'03* et *CAp'04*, ces deux articles ont été sélectionnés pour des publications en version longue dans des revues nationales :

- Sebban, M., & Suchier, H.M. (2003). Etude sur l'Amélioration du Boosting : Réduction de l'Erreur et Accélération de la Convergence *Journal électronique d'intelligence artificielle 6-46*.
- Janodet, J.C., Nock, R., Sebban, M. & Suchier, H.M. (2004). Adaptation du boosting à l'inférence grammaticale *via* l'utilisation d'un oracle de confiance. *Revue d'intelligence artificielle, 19 (4-5)* (pp 713–740).

3 Gestion du bruit avec OraBoost

Résumé : Ce chapitre traite du problème du bruit dans le cadre du boosting. Après avoir montré expérimentalement les effets du bruit en matière de sur-apprentissage et de vitesse de convergence, nous passons en revue les principales solutions existantes. Nous présentons ensuite un nouveau cadre d'adaptation du boosting aux données bruitées et non pertinentes, à partir duquel nous proposons l'algorithme OraBoost. Celui-ci suppose notamment l'accès à un oracle de confiance. Le processus de repondération incorporant les informations apportées par cet oracle est présenté, et les principales propriétés théoriques de convergence sont démontrées.

3.1 Introduction

Par la nature même des bases de données manipulées aujourd'hui, la gestion du bruit est devenu un des problèmes majeurs de l'apprentissage automatique. Par soucis de simplification de la terminologie employée, nous regrouperons dans cette partie sous le terme de "données bruitées" à la fois celles dont la représentation ou l'étiquette a été corrompue (par un processus aléatoire par exemple) et celles n'apportant pas d'information utile pour l'apprentissage, usuellement baptisées "données non pertinentes". La motivation de ce regroupement terminologique est liée au fait que la solution que nous présentons dans ce chapitre présente l'intérêt non négligeable de gérer ces deux catégories de données.

Le principe d'une méthode de boosting telle que AdaBoost étant basé sur l'augmentation exponentielle du poids des exemples mal appris lors des itérations précédentes, on conçoit aisément les effets dramatiques, en termes d'erreur et de vitesse de convergence, de la présence de données bruitées au sein de l'échantillon d'apprentissage. Les travaux de Dietterich (2000b) confirment ceci de manière

expérimentale. S'il faut distinguer le bruit sur la représentation des exemples et le bruit sur les étiquettes, remarquons qu'AdaBoost, de par son processus, est surtout sensible à la deuxième de ces deux catégories. Nous ne nous intéresserons donc dans la suite qu'à celle-ci, même s'il est à noter que certains travaux ont traité du bruit au niveau des attributs, tels les travaux de pondération ou de sélection d'attributs (Liu & Setiono, 1996 ; Howe & Cardie, 1999). Par ailleurs, les travaux en sélection de prototypes ont permis de mettre en avant plusieurs catégories d'exemples ayant un impact néfaste sur l'apprentissage d'une hypothèse (voir Fig. 3.1).

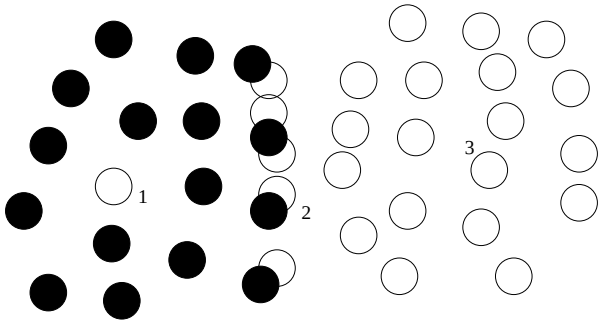


Fig. 3.1 – *Illustration des trois catégories principales d'exemples désignés comme étant nuisibles dans le domaine de la sélection de prototypes.*

La première correspond à celle des *outliers* (exemple 1 sur la figure). Il s'agit d'exemples, qui comme leur nom l'indique, ne sont pas représentatifs de la distribution statistique sous-jacente à leur classe. Ils peuvent donc prendre la forme d'exemples isolés au milieu d'autres exemples de la classe opposée.

La deuxième catégorie correspond aux exemples situés en *région bayésienne*, c'est à dire dans la zone de chevauchement entre deux classes (exemple 2). De tels exemples ne rajoutent ou n'enlèvent rien en termes d'erreur, puisque nous nous situons dans une zone d'indéterminisation totale, où l'erreur théorique est incompressible. Par contre, leur présence peut être nuisible en termes d'efficacité des algorithmes d'apprentissage.

Enfin, la troisième catégorie représente les exemples qui se trouvent au centre de clusters constitués d'instances d'une seule et même classe (exemple 3). Ces exemples étant très probablement bien classés, quelle que soit l'hypothèse construite, par ceux situés sur l'enveloppe convexe du cluster, ils peuvent être considérés comme inutiles, et leur suppression permet en général de réduire les contraintes de stockage de l'échantillon d'apprentissage.

Revenons maintenant sur les deux premières catégories d'exemples bruités. Par le biais d'une étude expérimentale, nous allons voir que leur présence peut

avoir non seulement un effet sur les performances du boosting en termes de taux d'erreur, mais également en termes de vitesse de convergence.

3.1.1 Boosting et sur-apprentissage

Pour prendre conscience de l'effet des données bruitées sur le taux d'erreur d'une hypothèse produite par AdaBoost, nous avons appliqué cet algorithme à un exemple artificiel à deux classes, *a priori* linéairement séparable, au sein duquel a été ajouté un niveau de bruit de 5% (en retournant l'étiquette des exemples). Nous avons utilisé comme apprenant faible un algorithme de construction de stumps. Comme nous l'avons déjà indiqué, ces derniers ont la particularité d'être des arbres à un seul niveau.

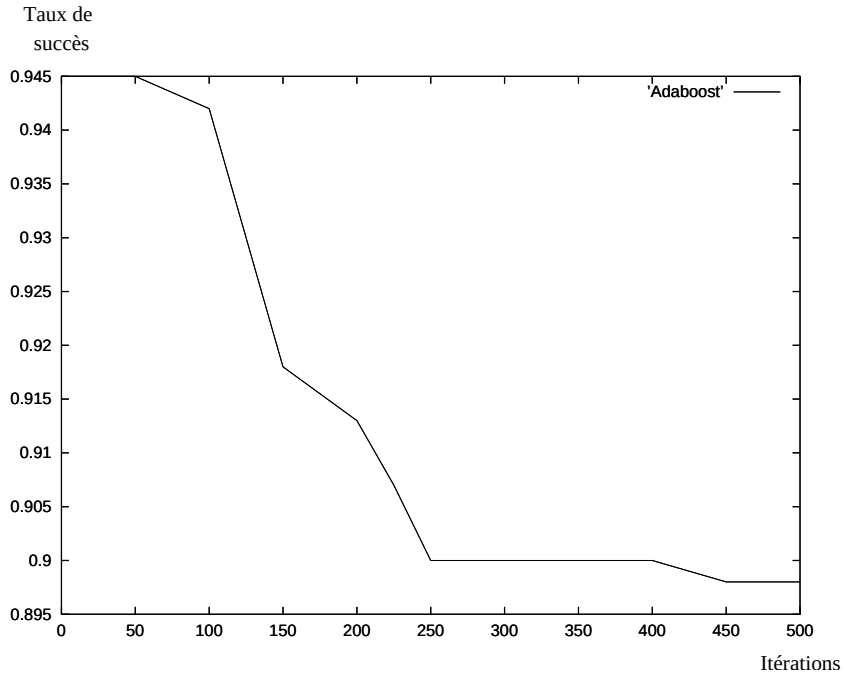


Fig. 3.2 – *L'évolution au cours des itérations de l'hypothèse forte produite par* AdaBoost.

La Figure 3.2 montre les résultats en termes de taux de succès estimé par validation croisée à 5 parties, sur les 500 premières itérations. Alors que le taux de succès optimal est atteint dès la première itération, on voit très clairement qu'AdaBoost diverge vite, confirmant le phénomène de sur-apprentissage sur des données bruitées.

Pour répondre à ce problème, certains travaux partent du constat que le principe de maximisation des marges sur l'échantillon d'apprentissage est responsable

du problème de sur-apprentissage du boosting en présence de données bruitées. Ces dernières ayant une marge négative, on assiste inexorablement à l'augmentation de leurs poids. Rätsch et al. (1999) proposent alors de changer la fonction d'optimisation des marges, en intégrant des *variables lâches* dans le processus de repondération. Celles-ci permettent de relâcher la contrainte d'apprentissage pour les exemples dont les poids sont trop élevés (les auteurs parlent alors de *marges douces*). Ainsi, un exemple accumulant des poids élevés lors des itérations 1 à $t-1$ se voit associer une variable de valeur élevée, ce qui permet de relâcher la contrainte de maximisation de sa marge. Si les auteurs présentent des expérimentations s'avérant prometteuses, aucune preuve théorique de convergence de l'erreur en généralisation n'est fournie.

D'autres travaux partent du fait que le processus de repondération exponentiel paraît favoriser le phénomène d'augmentation incontrôlée du poids des exemples bruités. Ils mettent donc en oeuvre un processus de repondération moins radical. Par exemple, Servedio (2001) propose un algorithme dans lequel la pondération est guidée par une marge minimale désirée θ pour chaque exemple. Supposons qu'à l'itération t, l'hypothèse h_t obtienne un taux d'erreur de $1/2-\gamma$. Pour chaque exemple, on calcule l'écart restant à la marge souhaitée $N_t(x_i) = N_{t-1}(x_i) + y_i h_t(x_i) - \theta$. La pondération s'effectue alors comme suit :

$$M_{t+1}(x_i) = \begin{cases} 1 & si \quad N_t(x_i) < 0 \\ (1-\gamma)^{N_t(x_i)/2} & si \quad N_t(x_i) \geq 0 \end{cases}$$

Ainsi, l'apprentissage est concentré sur les exemples dont la marge n'est pas estimée suffisamment élevée. Ce schéma de repondération garantit par ailleurs que le poids de chaque exemple reste dans un intervalle [0,1], évitant ainsi l'augmentation incontrôlée du poids de certains exemples. Il est à noter que les auteurs fournissent des résultats théoriques sur la convergence de cette méthode. Toutefois, ceux-ci dépendent du paramètre de marge θ, qui devra être correctement estimé en pratique.

Dans le cas de MAdaBoost (Domingo & Watanabe, 2000), l'opération consiste également en une modification de la fonction de repondération. Les auteurs ont choisi de borner le poids de chaque exemple par sa probabilité initiale, ce qui lui évitera de devenir arbitrairement trop grand. Bien que cette approche ait l'inconvénient de perdre les effets positifs de la fonction exponentielle sur les bornes, les auteurs prouvent néanmoins qu'il s'agit bien d'un algorithme de boosting.

Avec l'algorithme BrownBoost, Freund (1999) reprend une idées passée, établie dans un algorithme de boosting par vote majoritaire non adaptatif (Freund, 1995). Dans cette approche, l'algorithme y est optimisé pour minimiser l'erreur d'apprentissage en un nombre pré-fixé d'itérations k. L'objectif est que sur les k hypothèses faibles construites, au moins $p = k/2$ prédisent la bonne étiquette pour

chaque exemple d'apprentissage. Lorsqu'on arrive vers la fin des itérations, il est de moins en moins probable que les exemples ayant une marge courante fortement négative deviennent par la suite correctement étiquetés. De cette constatation, Freund tire la fonction de pondération suivante, qui n'est rien d'autre que la loi de probabilité d'une variable binomiale. Elle dépend du nombre d'itérations finales k, de l'itération courante i, du nombre de fois r où l'exemple a déjà été correctement étiqueté, et enfin de la probabilité de succès $1-\gamma$ imposée à toute hypothèse faible :

$$\alpha_i^r = \left(\begin{array}{c} k-i-1 \\ \lfloor \frac{k}{2} \rfloor - r \end{array} \right) (\frac{1}{2}+\gamma)^{\lfloor \frac{k}{2} \rfloor - r} (\frac{1}{2}-\gamma)^{\lfloor \frac{k}{2} \rfloor - i - 1 + r}.$$

Cette fonction est non monotone, et pour des marges faiblement négatives, ressemble à peu près à celle d'AdaBoost et sa fonction exponentielle. Par contre, au delà d'une certaine valeur, le poids diminue, surtout si on se trouve vers la fin des itérations. L'avantage de cette approche est que les outliers seront probablement détectés à un certain moment, et leur poids cessera alors d'augmenter. Cependant, ces outliers peuvent parfois être détectés (trop) tardivement, entraînant une influence négative sur l'hypothèse finale. De plus, cette stratégie nécessite de fixer *a priori* le nombre d'itérations, ce qui peut être une tâche délicate à réaliser en pratique.

3.1.2 Boosting et vitesse de convergence

Autant les effets négatifs du bruit sur les performances du boosting ont été souvent mentionnés dans la littérature ces dernières années, autant les causes d'une convergence tardive vers l'erreur optimale d'un algorithme d'apprentissage ont rarement été étudiées. Pourtant, nous pouvons mettre en avant les conséquences d'un fort chevauchement des densités de probabilités des classes à apprendre.

Nous avons exécuté AdaBoost sur un exemple artificiel présentant une erreur bayésienne de 20%. La Figure 3.3 représente l'évolution du taux de succès estimé par validation croisée sur les 500 premières itérations. On constate que le taux de succès optimal de 80%, *a priori* connu, est à peu près atteint seulement après 400 itérations. L'explication de ce phénomène est la suivante : les poids des exemples d'une classe se trouvant à la frontière bayésienne sont alternativement augmentés, car mal appris, puis baissés pour permettre le bon étiquetage des exemples de la classe opposée. Un tel phénomène oscillatoire freine le processus du boosting, et entraîne la création d'un ensemble redondant d'hypothèses faibles. Tout ceci a bien sûr un effet sur les performances de l'hypothèse globale, qui va en général demander plus d'itérations, et donc plus d'hypothèses faibles, pour converger vers son optimum.

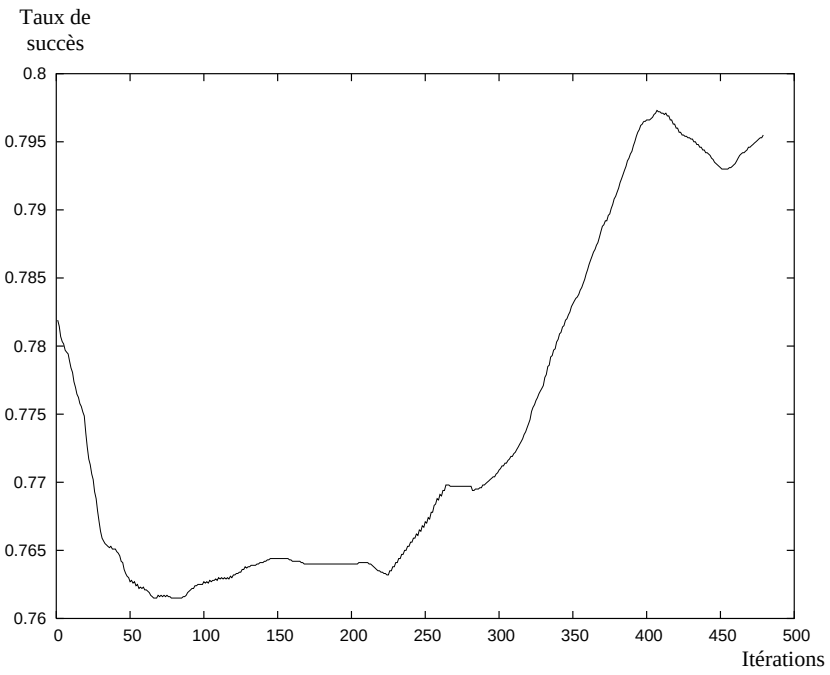


FIG. 3.3 – *Un exemple de convergence retardée par la présence de bruit bayésien.*

Pour tenir compte de ce phénomène, Nock et Sebban (2001) ont proposé une extension du Théorème 2.1 de Schapire et Singer (1999), présenté en Section 2.4.2, et bornant l'erreur empirique par le produit des Z_t. En définissant, pour toute représentation x_i dans l'échantillon d'apprentissage, les valeurs $n^+(x_i)$ et $n^-(x_i)$ comme étant respectivement le nombre d'exemples de classe $+1$ et -1 partageant la représentation x_i, ils définissent la quantité $\delta(x_i)$ comme suit :

$$\delta(x_i)) = \frac{n^+(x_i) - n^-(x_i)}{n^+(x_i) + n^-(x_i)}.$$

Le nouveau théorème s'énonce alors ainsi :

Théorème 3.1 *En utilisant* ADABOOST*, la borne suivante sur l'erreur en apprentissage de l'hypothèse globale H_T est atteinte :*

$$\hat{err}(H_t) = \frac{1}{m}|\{e_i \in E : H_T(x_i) \neq y_i\}| \leq (\prod_{t=1}^{T} Z_t) E_{D_{T+1}}[\delta(x)] + \epsilon^*$$

où ϵ^* est l'erreur minimale sur l'échantillon d'apprentissage, et $E_{D_{T+1}}[\delta(x)]$ est l'espérance de $\delta(x)$ sur la distribution D_{T+1} telle que

$$D_{T+1}(x_i) = \frac{e^{-y_i \sum_t \alpha_t h_t(x_i)}}{m \prod_t Z_t}.$$

Ce théorème permet d'intégrer dans la borne un terme correspondant à l'erreur bayésienne, et de montrer qu'une repondération, uniquement des exemples d'apprentissage non concernés par cette erreur, permet d'accélérer la vitesse de convergence.

Cependant, ce résultat est avant tout théorique. Il suppose en effet que les exemples concernés par cette erreur sont des exemples partageant exactement leur représentation avec des exemples de classe opposée. Or ce genre de situation n'est que peu fréquent dans la pratique, surtout lorsque le nombre d'attributs est élevé. La solution que nous proposons dans la section suivante va permettre notamment de supprimer cette contrainte forte.

3.2 Principes d'OraBoost

Si les approches de gestion du bruit présentées jusqu'ici semblent intéressantes au premier abord, elles présentent comme défaut, soit d'abandonner la fonction exponentielle, pourtant essentielle dans les propriétés théoriques de convergence, soit de nécessiter des paramètres estimant le niveau de bruit supposé de l'échantillon d'apprentissage.

Nous proposons ici une approche alternative, en faisant le choix de conserver la fonction de repondération exponentielle. De plus, nous souhaitons proposer un algorithme qui tienne aussi compte des problèmes de vitesse de convergence liés au bruit. Avant d'exposer notre approche, notons que puisqu'AdaBoost est sensible aux données bruitées, une solution de type *filter* consisterait à les supprimer en amont du processus de boosting, en utilisant par exemple une méthode de sélection de prototypes (Brodley & Friedl, 1996 ; Wilson & Martinez, 1997). Cette approche n'est pas forcément la plus judicieuse, ceci pour deux raisons principales. La première vient du fait que, comme l'ont montré de récents travaux en apprentissage semi-supervisé (Seeger, 2000), l'intérêt des descriptions des exemples, indépendamment de leur étiquette, n'est pas à négliger. La deuxième raison est liée au fait que supprimer à tort un exemple de l'échantillon d'apprentissage nous expose alors à une perte d'information, qui pourra avoir des conséquences non négligeables sur les performances de l'hypothèse finale.

En utilisant la terminologie employée en sélection de prototypes, nous avons fait le choix d'une solution de type *embedded* : plutôt que de supprimer les exemples jugés bruités, ceux-ci sont conservés dans l'échantillon d'apprentissage, mais nous supposons qu'une confiance peut être associée à leur étiquette. Ainsi, nous désirons que celle-ci soit maximale pour un exemple de centre de cluster, car celui-ci sera entièrement représentatif de la classe à laquelle il appartient (et par la même, du problème à apprendre). En revanche, la confiance associée à l'étiquette d'un outlier devra être minimale, un tel exemple n'étant que peu probablement repré-

sentatif de sa classe. De plus, les exemples situés en zone d'erreur bayésienne, impliqués dans les phénomènes de ralentissement de la convergence d'ADABOOST, devront se voir associer une confiance de valeur intermédiaire.

Pour prendre cette information en compte, nous avons mis en place une procédure de repondération dans laquelle intervient un *oracle de confiance*. Nous fournissons dans ce chapitre plusieurs résultats théoriques dans un tel contexte. Une valeur positive fournie par l'oracle pour un exemple signifie que nous pouvons être confiants dans sa classe. En revanche, une valeur négative signifie que l'on doit émettre des doutes sur son étiquetage. Il est crucial de remarquer qu'une confiance négative ne signifie donc pas que l'exemple appartient à la classe opposée, mais plutôt que l'on ne connaît pas sa classe réelle. De plus, il est nécessaire de supposer que l'oracle fournit des valeurs *réelles* dans $[-1, +1]$ plutôt que des valeurs *entières* dans $\{-1, +1\}$. Ce choix est lié au fait que cet oracle devra être simulé en pratique. Il est préférable de tenir compte des imperfections de cette simulation dès l'étude théorique.

Soit E un ensemble d'exemples de cardinal $|E| = m$. Chaque exemple e_i est composé d'un vecteur de représentation x_i et d'une étiquette y_i pouvant être soit positive ($y_i = +1$), soit négative ($y_i = -1$). L'oracle de confiance intervient ici sous la forme d'une fonction q, qui associe une confiance à tout exemple de E. Rappelons qu'une confiance positive signifie que e_i appartient certainement à la classe y_i qui lui est assignée, donc que e_i n'est probablement pas bruité, tandis qu'une confiance négative signifie que l'on ne peut rien supposer sur l'étiquette de e_i. Pour simplifier notre développement, nous supposerons que $q(e_i)$ n'est pas nul, ce qui signifie que l'oracle ne s'abstient pas, qu'il émet nécessairement un avis sur tous les exemples d'apprentissage ; cette hypothèse peut être levée moyennant un développement théorique plus important, proche de celui de (Schapire & Singer, 1999, p.8). D'autre part, les valeurs de $q(.)$ seront supposées constantes au cours du processus d'apprentissage.

Notre algorithme de boosting, que nous avons baptisé ORABOOST (voir pseudo-code dans l'Algorithme 3) suit le même schéma qu'ADABOOST : on définit une première distribution $\mathbf{w}_0$ uniforme sur l'échantillon d'apprentissage E ; ensuite, à chaque itération t, on utilise l'apprenant faible WL sur E et $\mathbf{w}_t$ pour obtenir une hypothèse faible h_t et on repondère les exemples d'apprentissage en construisant la distribution $\mathbf{w}_{t+1}$; enfin, au bout de T itérations, on combine les T hypothèses faibles produites en une hypothèse globale H_T "forte".

Cependant, le schéma de repondération des exemples change sensiblement, dans la mesure où il fait intervenir explicitement les réponses de l'oracle. Par cette mise à jour, nous désirons non seulement augmenter le poids de tout exemple mal classé pour lequel l'oracle se dit confiant, mais également baisser le poids de tout exemple "douteux", qu'il soit bien ou mal classé par l'hypothèse courante. La règle de mise à jour que nous proposons permet de réaliser cela. En effet, pour un

Algorithme 3: *Pseudo-code d'*ORABOOST

Données : Un échantillon E, un apprenant faible WL, un oracle q,
Retourner : Une combinaison H_T d'hypothèses.
Pour tout $e_i \in E$ **faire**
  $w_0(e_i) \longleftarrow 1/|E|$;
Finpour
Pour tout $t = 0$ **jusqu'à** T **faire**
  $h_t \longleftarrow \text{WL}(E, \mathbf{w}_t)$;
  $c_t \longleftarrow$ `calcul_coeff_boosting`();
  Pour tout $i = 0$ **jusqu'à** $|E|$ **faire**
    si $(q(e_i) < 0)$ **alors**
    $a_t(e_i) \longleftarrow -q(e_i)$ **sinon**
    $a_t(e_i) \longleftarrow q(e_i) y_i h_t(x_i)$;
  Finpour
  $Z_t \longleftarrow \sum_{e_i \in E} w_t(e_i) \exp(-c_t a_t(e_i))$;
  Pour tout $i = 0$ **jusqu'à** $|E|$ **faire**
    $w_{t+1}(e_i) \longleftarrow w_t(e_i) \exp(-c_t a_t(e_i))/Z_t$;
  Finpour
Finpour
Retourner H_T **tel que** $H_T(\omega) = \text{signe}\left(\sum_{t=0}^{T} c_t h_t(\omega)\right)$;

exemple e_i à qui l'oracle attribue une confiance positive ($q(e_i) > 0$) et qui n'est donc probablement pas bruité, la repondération s'effectue comme suit :

$$w_{t+1}(e_i) \quad = \quad \frac{w_t(e_i) \exp(-c_t q(e_i) y_i h_t(x_i))}{Z_t}.$$

Pour un exemple e_i de confiance négative ($q(e_i) < 0$), donc un exemple suspect, on a :

$$w_{t+1}(e_i) \quad = \quad \frac{w_t(e_i) \exp(c_t q(e_i))}{Z_t}.$$

En prouvant que c_t (renvoyée par la fonction `calcul_coeff_boosting()`, qui sera discutée dans la suite) est toujours supérieure à 0, ce qui sera fait dans le Lemme 3.2, nous pourrons alors assurer que les objectifs visés seront bien atteints.

Une autre modification majeure intervient dans notre nouvel algorithme de boosting. Elle concerne ce fameux coefficient c_t, différent de celui d'ADABOOST et qui nécessite d'être calculé optimalement pour préserver les propriétés théoriques du boosting. Nous développerons dans ce qui suit deux approches, aboutissant toutes les deux à une même contrainte $\mathcal{C}$ qui caractérise l'optimalité du choix de c_t. La première, développée dans la Section 3.3.1, consiste à montrer que notre

règle de mise à jour des poids est la solution optimale d'un problème de minimisation d'une fonction convexe (divergence de Breigman), quand on choisit correctement c_t. La seconde (Section 3.3.2), plus classique, montre qu'on peut minorer l'erreur en apprentissage de H_T en choisissant cette même valeur optimale de c_t. La contrainte $\mathcal{C}$ est en fait une équation dont c_t est l'unique solution. Dans le cas d'ADABOOST, cette équation peut être résolue littéralement (en fonction des ϵ_t). Ce n'est pas le cas ici, car notre règle de mise à jour des poids utilise explicitement l'oracle, c'est-à-dire une fonction à valeurs réelles $q(.)$. Néanmoins, nous pourrons obtenir une valeur approchée $\hat{c}_t$ de la valeur théorique de c_t avec une précision quelconque. Pour ce faire, nous commencerons par expliciter une hypothèse d'apprentissage faible, propre au boosting avec oracle (Section 3.3.3), ce qui permettra ensuite d'approcher c_t par dichotomie (Section 3.3.4).

3.3 Résultats théoriques

3.3.1 Sur la règle de mise à jour des poids

Soit $\mathcal{P}_m$ l'espace des vecteurs de probabilité de dimension m et $\mathbf{u} \in \mathcal{P}_m$ le vecteur uniforme. Nous supposons l'existence d'un vecteur $\mathbf{w}_t \in \mathcal{P}_m$ qui contient le poids de chaque exemple de E à l'étape t, avec $\mathbf{w_0} = \mathbf{u}$ (la distribution initiale étant donc uniforme, comme dans ADABOOST).

Nous savons, depuis (Kivinen & Warmuth, 1999a), que le boosting standard tel que défini dans (Schapire & Singer, 1999) est un cas particulier de l'optimisation sous contrainte d'une divergence de Breigman. En replaçant ce problème dans notre cadre, nous établissons l'équation que doit satisfaire le coefficient c_t optimal.

La stratégie du boosting standard est de calculer successivement $\mathbf{w}_{t+1}$ à partir de $\mathbf{w}_t$ en minimisant la *divergence d'information*

$$\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1},\mathbf{w}_t) \;=\; \sum_{e_i=(x_i,y_i)\in E} i(\mathbf{w}_{t+1},\mathbf{w}_t)(e_i),$$

où $\mathbf{i}(.,.)$ est le vecteur de composante

$$i(\mathbf{w}_{t+1},\mathbf{w}_t)(e_i) \;=\; w_{t+1}(e_i)\ln\left(\frac{w_{t+1}(e_i)}{w_t(e_i)}\right) - w_{t+1}(e_i) + w_t(e_i),$$

pour tout $e_i \in E$. Remarquons que cette divergence d'information est une fonction convexe de $\mathbf{w}_{t+1}$.

Nous souhaitons optimiser $\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1},\mathbf{w}_t)$ sous deux contraintes. La première est immédiate ; elle exprime le fait que $\mathbf{w}_{t+1}$ est une distribution, ce qui ne change

évidemment pas dans notre nouveau schéma :

$$\mathbf{1}.\mathbf{w}_{t+1} - 1 \;=\; 0. \tag{3.1}$$

La seconde contrainte intègre une fonction de perte pour chaque exemple e_i de E. Dans le boosting classique, elle exprime le fait que la dernière hypothèse apprise est, en quelque sorte, la plus mauvaise hypothèse sur la nouvelle distribution, ses performances étant en moyenne équivalentes à celles de la décision aléatoire :

$$\mathbf{w}_{t+1}.(\mathbf{y}\mathbf{h}_t) = 0 \;\Leftrightarrow\; \sum_{e_i \in E} w_{t+1}(e_i) y_i h_t(x_i) = 0.$$

Cette contrainte, combinée avec l'*hypothèse d'apprentissage faible* (Kearns & Vazirani, 1994), assure que la nouvelle hypothèse h_{t+1} construite sur la distribution $\mathbf{w}_{t+1}$ est significativement différente de h_t, ce qui impose à l'algorithme d'apprendre quelque chose de nouveau sur E, à chaque étape.

Dans notre cas, cette contrainte est différente. On commence par partitionner E en deux sous-ensembles, l'ensemble $E_N = \{e_i \in E : q(e_i) < 0\}$ des exemples dont l'étiquette est douteuse, et l'ensemble $E_{\overline{N}} = E \backslash E_N$ des exemples probablement non bruités, et on pose :

$$\begin{aligned} \mathbf{w}_{t+1}.\mathbf{a}_t &= 0, \text{ avec} \\ \forall e_i \in E_{\overline{N}}, a_t(e_i) &= q(e_i) y_i h_t(x_i) \text{ et} \\ \forall e_i \in E_N, a_t(e_i) &= -q(e_i). \end{aligned} \tag{3.2}$$

Pour faire une comparaison avec la seconde contrainte du boosting classique, notons que l'égalité (3.2) entraîne :

$$\sum_{e_i \in E_{\overline{N}}} w_{t+1}(e_i) q(e_i) y_i h_t(x_i) \;=\; \sum_{e_i \in E_N} w_{t+1}(e_i) q(e_i).$$

Comme $\forall e_i \in E_N$, $q(e_i) < 0$, le membre droit de cette équation est strictement négatif, ce qui rend la dernière hypothèse construite h_t particulièrement mauvaise sur $E_{\overline{N}}$, avec une distribution dépendant à la fois de $\mathbf{w}_{t+1}$ et de $\mathbf{q}$. On voit ici clairement la différence entre notre schéma de repondération et celui du boosting classique qui assure une performance moyenne (comparable au choix aléatoire), uniquement avec la distribution $\mathbf{w}_{t+1}$ sur l'ensemble E tout entier.

La minimisation de la divergence d'information sous les contraintes (3.1) et (3.2) est obtenue comme solution ($\forall e_i \in E$) de :

$$\left[\partial_{\mathbf{w}_{t+1}} \mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t) + b_t \partial_{\mathbf{w}_{t+1}} (\mathbf{1}.\mathbf{w}_{t+1} - 1) + c_t \partial_{\mathbf{w}_{t+1}} (\mathbf{w}_{t+1}.\mathbf{a}_t)\right](x_i) = 0, \tag{3.3}$$

où b_t et c_t sont des multiplicateurs de Lagrange. La résolution de (3.3) nous permet d'obtenir, pour tout $e_i \in E$:

$$\left(\ln \frac{\mathbf{w}_{t+1}}{\mathbf{w}_t} + b_t \mathbf{1} + c_t \mathbf{a}_t\right)(x_i) = 0 \quad \Leftrightarrow \quad w_{t+1}(e_i) = \frac{w_t(e_i)\exp(-c_t a_t(e_i))}{\exp(b_t)}.$$

La contrainte (3.1) permet de déduire :

$$b_t \quad = \quad \ln \sum_{e_i \in E} w_t(e_i)\exp(-c_t a_t(e_i)). \tag{3.4}$$

Le terme dans le ln(.) correspond donc au coefficient de normalisation pour $\mathbf{w}_{t+1}$:

$$Z_t \quad = \quad \sum_{e_i \in E} w_t(e_i)\exp(-c_t a_t(e_i)). \tag{3.5}$$

La dernière inconnue c_t est obtenue par la contrainte (3.2) comme solution de :

$$\sum_{x_i \in E} w_t(e_i) a_t(x_i)\exp(-c_t a_t(e_i)) \quad = \quad 0. \tag{3.6}$$

3.3.2 Résultat sur l'erreur en apprentissage

La valeur de c_t est donc obtenue comme solution de l'équation (3.6). On peut remarquer que cette équation est strictement équivalente à :

$$\left(\frac{\partial Z_t}{\partial c_t}\right) \quad = \quad 0.$$

Comme Z_t est une fonction convexe de c_t, résoudre (3.6) revient donc à chercher le coefficient c_t qui minimise Z_t. Ce problème de minimisation est intimement lié avec celui de la minimisation de l'erreur en apprentissage commise par H_T.

Commençons par définir cette erreur : puisque l'étiquette des exemples de E_N n'est pas fiable, nous définissons l'erreur empirique de notre algorithme de boosting sur $E_{\overline{N}}$ uniquement. Aussi, on pose $\forall T > 0$,

$$\hat{err}(E_{\overline{N}}, H_T) \quad = \quad \frac{1}{|E_{\overline{N}}|} \sum_{e_i \in E_{\overline{N}}} [\![y_i \neq H_T(x_i)]\!], \tag{3.7}$$

où $[\![\pi]\!]$ désigne la valeur de vérité d'un prédicat π (dans {0,1}).

Nous obtenons le résultat suivant :

Lemme 3.1

$$\hat{err}(E_{\overline{N}}, H_T) \quad \leq \quad \frac{m}{|E_{\overline{N}}|} \left(\prod_{t=0}^{T} Z_t\right).$$

Preuve : comme $\forall e_i \in E_{\overline{N}}$, $q(e_i) > 0$, nous avons pour tous ces exemples :

$$[\![H_T(x_i) \neq y_i]\!] \quad \leq \quad \exp(-q(e_i)y_i \sum_{t=0}^{T} c_t h_t(x_i)).$$

D'autre part, en déroulant la règle de mise à jour, on obtient, $\forall e_i \in E_{\overline{N}}$:

$$w_{T+1}(e_i) \quad = \quad \frac{w_0(e_i) \exp(-q(e_i)y_i \sum_{t=0}^{T} c_t h_t(x_i))}{\left(\prod_{t=0}^{T} Z_t\right)}.$$

Par conséquent,

$$[\![H_T(x_i) \neq y_i]\!] \leq m w_{T+1}(e_i)(\prod_{0 \leq t \leq T} Z_t).$$

En sommant pour tout $e_i \in E_{\overline{N}}$, nous obtenons

$$\hat{err}(E_{\overline{N}}, H_T) \leq (\frac{m}{|E_{\overline{N}}|})(\prod_{0 \leq t \leq T} Z_t)(\sum_{e_i \in E_{\overline{N}}} w_{T+1}(e_i)).$$

Enfin, en utilisant le fait que $\sum_{e_i \in E_{\overline{N}}} w_{T+1}(e_i) \leq 1$, nous obtenons le Lemme. □

Comme le terme $(m/|E_{\overline{N}}|)$ est constant pour E et $E_{\overline{N}}$, le Lemme 3.1 signifie que pour minimiser l'erreur sur $E_{\overline{N}}$, il faut se concentrer sur la minimisation de chaque Z_t. Le constat est désormais clair : résoudre (3.6) nous permet d'obtenir non seulement la solution de (3.3), mais également la minimisation de l'erreur sur $E_{\overline{N}}$ au regard du Lemme 3.1.

3.3.3 Une hypothèse d'apprentissage faible adaptée

Pour trouver une solution de (3.6), il nous faut maintenant formuler une hypothèse :

$\forall t \geq 0$, il existe une constante $\gamma_t > 0$ telle que :

$$\frac{\sum_{e_i \in E_{\overline{N}}} (w_t(e_i) q(e_i) y_i h_t(x_i)}{\sum_{e_i \in E_{\overline{N}}} w_t(e_i) q(e_i)} \quad = \quad \gamma_t. \tag{3.8}$$

En suivant la terminologie classique du boosting, (3.8) doit être vue comme une hypothèse d'apprentissage faible (*Weak Learning Assumption*, WLA). Notons que si h_t correspond au choix aléatoire, alors $\gamma_t = 0$. En d'autres termes, (3.8) traduit le fait que h_t doit être légèrement plus performante que le choix aléatoire pour pouvoir être considérée comme une hypothèse faible.

Notre WLA est différente de celle du boosting classique (Kearns & Vazirani, 1994) pour deux raisons. Tout d'abord, elle est locale : elle se concentre sur un

sous-ensemble de E. Ensuite, elle s'appuie sur une distribution qui n'est pas exactement $\mathbf{w}_t$ mais une distribution $\mathbf{w}_{q,t} \in \mathcal{P}_m$ paramétrée par q qu'on définit ainsi, $\forall e_i \in E$:

$$w_{q,t}(e_i) \;=\; \frac{|q(e_i)|w_t(e_i)}{Q_t}, \text{ avec } Q_t = \sum_{e_j \in E} |q(e_j)|w_t(e_j).$$

Notons que $\mathbf{w}_{q,t} \in \mathcal{P}_m$, car tous les exemples ont une confiance $q(.)$ différente de 0. Nous étendons cette définition en posant, pour tout $E' \subseteq E$,

$$W_{q,t}(E') \;=\; \sum_{e_i \in E'} w_{q,t}(e_i).$$

Afin de compléter la discussion autour de la contrainte 3.2 que nous avons commencée dans la section 3.3.1, nous définissons

$$\begin{aligned} E^+_{\overline{N},t} &= \{e_i \in E_{\overline{N}} : y_i h_t(x_i) = +1\}, \text{ et} \\ E^-_{\overline{N},t} &= \{e_i \in E_{\overline{N}} : y_i h_t(x_i) = -1\}. \end{aligned}$$

La WLA s'écrit alors :

$$W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) \;=\; \gamma_t W_{q,t}(E_{\overline{N}}). \tag{3.9}$$

Nous ne pouvons donc pas choisir $h_{t+1} = h_t$ sous la WLA, car sinon (3.2) impliquerait que $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) < 0$, tandis que (3.9) impliquerait $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) > 0$. Comme dans le cadre classique d'ADABOOST, notre WLA force donc l'algorithme d'apprentissage à retourner une hypothèse h_{t+1} différente de h_t sur $E_{\overline{N}}$, donc à apprendre quelque chose de nouveau.

3.3.4 Approximation des coefficients c_t

Sous notre WLA, nous obtenons le résultat suivant :

Lemme 3.2 *La solution de (3.6) est unique, strictement positive, et elle vérifie :*

$$\frac{1}{2\overline{q}} \ln\left(\frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})}\right) \leq c_t \leq \frac{1}{2\underline{q}} \ln\left(\frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})}\right),$$

où $\underline{q} = \min_{e_i \in E} |q(e_i)|$ *et* $\overline{q} = \max_{e_i \in E} |q(e_i)|$.

En d'autres termes, $c_t = r\ln((1 - W_{q,t}(E^-_{\overline{N},t}))/W_{q,t}(E^-_{N,t}))$ est solution de l'équation (3.6) pour un certain $r \in [1/2\overline{q},1/2\underline{q}]$. Cette borne est plus précise que celle donnée dans (Schapire & Singer, 1999) (où $c_t \in \mathbb{R}$).

De plus, elle permet une approximation très rapide de c_t par dichotomie. Supposons qu'on veuille approximer c_t par un $\hat{c}_t$ tel que $|\hat{c}_t - c_t|/|c_t| < \epsilon$ pour un $0 < \epsilon \leq 1$. L'approximation dichotomique se fera alors en au plus $\mathcal{O}(\ln(\overline{q}/\underline{q}) + \ln(1/\epsilon))$ étapes. Cette efficacité est d'autant plus appréciable que ce calcul devra être fait à chaque itération de boosting, soit T fois.

Preuve : posons

$$g(c) = \sum_{e_i \in E} (w_t(e_i)a_t(e_i)\exp(-ca_t(e_i)). \tag{3.10}$$

$g'(c) = -\sum_{e_i \in E} (w_t a_t^2)(e_i)\exp(-ca_t(x_i)) < 0$. Donc $g(c)$ est strictement décroissante sur $\mathbb{R}$. Comme $\lim_{c\to+\infty} g(c) = -\infty$ (à la condition qu'il existe $e_i \in E$ tel que $a_t(x_i) < 0$) et $\lim_{c\to-\infty} g(c) = +\infty$ (à la condition qu'il existe $e_i \in E$ tel que $a_t(x_i) > 0$), l'équation (3.6) admet une solution unique c_t. De plus,

$$\begin{aligned} \frac{g(0)}{Q_t} &= W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) + W_{q,t}(E_N) \\ &= \gamma_t W_{q,t}(E_{\overline{N}}) + W_{q,t}(E_N) \\ &= \gamma_t + (1 - \gamma_t)W_{q,t}(E_N). \end{aligned} \tag{3.11}$$

Sous la WLA, on a donc $g(0) > 0$, et comme g est strictement décroissante, on en déduit que $c_t > 0$. Enfin,

$$\begin{aligned} \frac{g(c_t)}{Q_t} = 0 = {} & \textstyle\sum_{e_i \in E^+_{\overline{N},t} \cup E_N} w_{q,t}(e_i)\exp\left(-c_t|q(e_i)|\right) \\ & - \textstyle\sum_{e_i \in E^-_{\overline{N},t}} w_{q,t}(e_i)\exp\left(c_t|q(e_i)|\right). \end{aligned}$$

Comme $\underline{q} \leq |q(e)| \leq \overline{q}$ pour tout $e \in E$, nous obtenons :

$$\begin{aligned} W_{q,t}(E^+_{\overline{N},t} \cup E_N)e^{-c_t\overline{q}} - W_{q,t}(E^-_{\overline{N},t})e^{c_t\overline{q}} \leq 0 \quad \text{et} \\ W_{q,t}(E^+_{\overline{N},t} \cup E_N)e^{-c_t\underline{q}} - W_{q,t}(E^-_{\overline{N},t})e^{c_t\underline{q}} \geq 0. \end{aligned}$$

Comme $W_{q,t}(E^+_{\overline{N},t} \cup E_N) = 1 - W_{q,t}(E^-_{\overline{N},t})$, on en déduit le résultat. □

3.3.5 Convergence théorique de l'erreur en apprentissage

Sous notre WLA, nous pouvons maintenant établir le lemme suivant :

Lemme 3.3 *$\forall t \geq 0$, en posant $\sigma_t = \gamma_t \frac{\underline{q}}{\overline{q}}$, on a :*

$$Z_t \leq \sqrt{1-\sigma_t^2} < \exp\left(-\frac{\sigma_t^2}{2}\right).$$

Comme nous l'avons évoqué au début de cette section, nous supposons que la fonction $q(.)$, donc $\underline{q}$ et $\overline{q}$, sont des constantes au cours des itérations du boosting. Aussi, sous la WLA, la valeur maximale de Z_t est toujours inférieure à 1. Or comme $e\hat{r}r(E_{\overline{N}},H_T) \leq \frac{m}{|E_{\overline{N}}|}\left(\prod_{t=0}^{T} Z_t\right)$, ceci nous garantit une convergence exponentielle de l'erreur en apprentissage de H_T vers 0.

Preuve : on a l'inégalité suivante, $\forall x \in [-1,1], \forall \eta \in \mathbb{R}$:

$$\exp(-\eta x) \leq \frac{1+x}{2}\exp(-\eta) + \frac{1-x}{2}\exp(\eta). \tag{3.12}$$

En effet, la fonction $x \mapsto \exp(-\eta x)$ est convexe et le membre droit de l'inégalité est l'équation de la droite passant par les points de coordonnées $(-1, \exp(\eta))$ et $(1, \exp(-\eta))$. En posant $\eta = c\overline{q}$ et $x_i = a_t(x_i)/\overline{q}$, nous obtenons $\forall e_i \in E$:

$$\exp(-c a_t(x_i)) \leq \frac{\overline{q}+a_t(e)}{2\overline{q}}\exp(-c\overline{q}) + \frac{\overline{q}-a_t(e)}{2\overline{q}}\exp(c\overline{q}). \tag{3.13}$$

En appelant $\ell(e_i,c)$ le membre droit de l'Inéquation (3.13), nous obtenons donc :

$$Z_t \leq \inf_{c\in\mathbb{R}} \sum_{e_i \in E} w_t(e_i)\ell(e_i,c). \tag{3.14}$$

Le c minimisant le membre droit de l'Inéquation (3.14) est

$$c = \frac{1}{2\overline{q}} \ln\left(\frac{\overline{q} + \sum_{e_i\in E}(w_t a_t)(e_i)}{\overline{q} - \sum_{e_i\in E}(w_t a_t)(e_i)}\right) = \frac{1}{2\overline{q}} \ln\left(\frac{\overline{q}+g(0)}{\overline{q}-g(0)}\right),$$

où $g(.)$ est la fonction définie dans l'Equation (3.10). En remplaçant c par cette valeur dans l'Inéquation (3.14), on obtient :

$$Z_t \leq \sqrt{1-\left(\frac{g(0)}{\overline{q}}\right)^2}, \tag{3.15}$$

Enfin, en utilisant le fait que $g(0) \geq \gamma_t Q_t$ (d'après l'équation (3.11)) et que $Q_t \geq \underline{q}$, nous obtenons le résultat du lemme. □

3.3.6 Convergence, en pratique, de l'erreur en apprentissage

La borne du Lemme 3.3 est purement théorique, dans la mesure où c_t ne peut pas être calculé de manière exacte, mais seulement approché par une valeur $\hat{c}_t$, calculée par dichotomie. Aussi, la valeur exacte de Z_t ne peut pas non plus être calculée, mais seulement approchée par une valeur $\hat{Z}_t = \sum_{e \in E} w_t(e) \exp(-\hat{c}_t a_t(e))$. Or une approximation trop grossière de $\hat{c}_t$ ne permettra certainement pas d'assurer que $\hat{Z}_t < 1$, donc la convergence de notre schéma de boosting en pratique pourrait être remis en cause. Notre dernier résultat montre que $\hat{Z}_t < 1$ dès que $|\hat{c}_t - c_t| < \exp\left(-\frac{\sigma_t^2}{2}\right)$. Nous obtenons plus précisément le résultat suivant :

Lemme 3.4 $\forall t \geq 0$,

$$si\ |\hat{c}_t - c_t| \leq \frac{(\ln 2)\sigma_t^2}{4\overline{q}_t},\ alors\ \hat{Z}_t \leq \sqrt{1 - \frac{\sigma_t^2}{2}} < \exp\left(-\frac{\sigma_t^2}{4}\right).$$

Preuve : soit $\zeta(y) = \sum_{e_i \in E} w_t(e_i) \exp(-y a_t(x_i))$. En développant ζ en séries entières, on obtient $\forall y \in \mathbb{R}$,

$$\zeta(y) = \sum_{k=0}^{+\infty} \frac{\partial^k \zeta}{\partial y^k}(c_t) \frac{(y - c_t)^k}{k!} \leq \sum_{k=0}^{+\infty} \left| \frac{\partial^k \zeta}{\partial y^k}(c_t) \right| \frac{|y - c_t|^k}{k!} \ .$$

Comme $\left| \frac{\partial^k \zeta}{\partial y^k}(c_t) \right| \leq \overline{q}_t^k Z_t$ pour tout $k \geq 0$, on en déduit

$$\zeta(y) \leq Z_t \left(\sum_{k=0}^{+\infty} \frac{\overline{q}_t^k |y - c_t|^k}{k!} \right) = Z_t \exp\left(\overline{q}_t |y - c_t|\right) \ ,$$

donc $\hat{Z}_t \leq Z_t \exp\left(\overline{q}_t |\hat{c}_t - c_t|\right)$. Clairement, $\forall \epsilon > 0$, si $|\hat{c}_t - c_t| \leq \ln(1 + \epsilon)/\overline{q}_t$, alors $\hat{Z}_t \leq (1 + \epsilon) Z_t$. Donc en posant $\epsilon = \frac{\sigma_t^2}{4}$ et en utilisant le Lemme 3.3, on obtient

$$\hat{Z}_t \ \leq \ \sqrt{1 - \frac{\sigma_t^2}{2}} \tag{3.16}$$

Pour finir, on peut remarquer que $\ln(1 + \epsilon) \geq (\ln 2)\epsilon$ pour tout $\epsilon \in [0,1]$, du fait de la concavité de la fonction $x \mapsto \ln(1 + x)$. Donc l'Inéquation (3.16) est satisfaite dès que $|\hat{c}_t - c_t| \leq \frac{(\ln 2)\sigma_t^2}{4\overline{q}_t}$. □

3.4 Conclusion

Nous avons mis en place dans ce chapitre le cadre théorique d'une méthode de boosting résistante aux données bruitées. Cette méthode repose sur la mise en évidence de trois catégories d'exemples, relativement à leur impact sur les performances d'une méthode classique de boosting telle que AdaBoost : les outliers, qui peuvent dégrader les performances de l'hypothèse finale, les exemples situés en zone d'erreur bayésienne, qui peuvent ralentir la convergence vers l'erreur optimale, et ceux situés dans les centres de clusters, qui n'apportent pas d'information supplémentaire dans le processus d'apprentissage.

En supposant avoir accès, *via* un oracle, à une information sur la confiance d'un exemple, la méthode que nous proposons consiste à réduire invariablement le poids des exemples appartenant aux deux premières catégories. Ainsi, leur impact sur les performances de l'hypothèse finale est limité, voire annihilé. Les exemples de centre de clusters se voient quand à eux pondérés de la même manière qu'avec AdaBoost, qui tend également à diminuer leurs poids. Sous condition d'une nouvelle hypothèse d'apprentissage faible, nous avons fourni plusieurs résultats théoriques prouvant que notre algorithme OraBoost est bien un algorithme de boosting.

Il s'agit à présent de mettre notre solution en pratique. C'est l'objet du chapitre suivant, qui propose notamment une approche généraliste de mise en oeuvre d'un oracle, et l'application de celle-ci dans le cas de données numériques d'une part, et de données symboliques d'autre part.

4 Application d'OraBoost aux données numériques et symboliques

Résumé : Ce chapitre présente une mise en pratique du modèle théorique présenté précédemment. Cette mise en oeuvre nécessitant l'utilisation d'un oracle de confiance, nous présentons tout d'abord une simulation de celui-ci. Ensuite, nous appliquons notre modèle au cas de données numériques, avant de l'exploiter dans le cadre de l'inférence grammaticale, domaine manipulant des séquences de caractères.

4.1 Simulation d'un oracle de confiance

Nous nous intéressons dans cette section à la mise en oeuvre d'un oracle de confiance. En pratique, nous pouvons assimiler celui-ci à un expert du domaine étudié capable de certifier ou non la légitimité de l'étiquette d'un exemple. Son existence est tout à fait crédible dans les domaines où le niveau d'expertise est suffisant. Ceci n'est malheureusement pas toujours le cas, et afin de pouvoir appliquer notre schéma de boosting sur un large spectre de problèmes réels, nous allons devoir parfois simuler le comportement d'un oracle. Dans cette section, nous décrivons une approche empirique permettant d'effectuer cette simulation, avec comme objectif principal de s'assurer qu'une confiance positive donnée par l'oracle correspond à un exemple dont l'étiquette est très probablement correcte. Nous proposons ici une stratégie qui pourrait s'apparenter aux méthodes dites de *co-training* (Blum & Mitchell, 1998). Ces méthodes sont utilisées lorsque l'on dispose d'une certaine quantité d'exemples, dont seule une petite proportion est étiquetée. On suppose par ailleurs qu'il existe une partition naturelle des attributs des exemples en deux sous-ensembles d'attributs, ce qui revient à considérer

que l'on dispose de deux représentations différentes des données. Un algorithme d'apprentissage auxiliaire apprend un modèle à partir d'une des deux représentations des exemples étiquetés. Ce modèle est alors utilisé pour étiqueter les exemples qui ne le sont pas encore. Un algorithme d'apprentissage utilise ensuite le deuxième sous-ensemble d'attributs, et la totalité des exemples, désormais tous étiquetés, pour construire un modèle final. Dans notre cas, l'oracle joue le rôle de l'algorithme d'apprentissage auxiliaire. Cependant, il exploite la même représentation des données que l'algorithme principal. De plus, plutôt que d'étiqueter des exemples qui ne le sont pas, il fournit une information supplémentaire sur la confiance en l'étiquette des exemples. Nous proposons ici une stratégie basée sur un graphe de voisinage, qui permet de distinguer géométriquement les exemples douteux des exemples sûrs.

Nous pensons que les graphes des k plus proches voisins (kPPV) (Cover & Hart, 1967) sont particulièrement adaptés pour traiter un tel problème. Rappelons qu'une hypothèse de kPPV attribue la classe d'un exemple e_i en calculant la classe majoritaire parmi les k exemples de E qui sont les plus proches de e_i. Cette méthode, au-delà du fait que son erreur limite est théoriquement bornée par deux fois l'erreur bayésienne, est très résistante au bruit. En effet, un exemple bruité isolé au milieu d'exemples de la classe opposée a un impact très limité sur la règle de classification. La propriété duale que l'on peut déduire de la remarque précédente, est qu'un exemple bruité peut être plus facilement détecté en analysant son voisinage. Si ses performances en généralisation, en tant qu'hypothèse d'apprentissage, sont une bonne indication des capacités d'une hypothèse de kPPV, elles ne seront pas au centre de notre étude. Le but est plutôt ici d'exploiter au mieux ce graphe comme oracle pour optimiser les performances du boosting.

Définissons la fonction de marge $m(e_i)$ d'un exemple e_i de classe y_i calculée à l'aide d'un graphe des kPPV :

$$m(e_i) \quad = \quad \frac{N_{y_i} - N_{\overline{y_i}}}{N_{y_i} + N_{\overline{y_i}}},$$

où N_{y_i} (resp. $N_{\overline{y_i}}$) désigne le nombre d'exemples parmi les k plus proches voisins de e_i étant de la même classe y_i (resp. étant de la classe opposée $\overline{y_i}$). Un exemple n'ayant que des voisins de la classe opposée à la sienne (resp. de la même classe) recevra une marge de -1 (resp. $+1$). Même si $m(e_i)$ semble représenter d'une certaine manière la valeur de confiance $q(e_i)$ utilisée dans le chapitre précédent, poser $q(e_i) = m(e_i)$ ne serait pas judicieux. En effet, la règle de classification d'un kPPV assure simplement la minimisation de l'erreur empirique, et ce n'est pas ce qui nous intéresse ici. Nous cherchons à fournir une mesure pertinente de la confiance des exemples, en nous assurant que les valeurs positives de $q(e_i)$ correspondent aux e_i ayant une étiquette correcte, dans $(100-\epsilon)$ % des cas (avec ϵ très petit). Or dans certains cas, même si $m(e_i)$ est légèrement plus grande

que 0, cette condition est loin d'être respectée (bien que la marge soit positive). Dans d'autres cas en revanche, il est possible qu'un exemple e_i de marge $m(e_i)$ négative mérite une confiance $q(e_i)$ positive. Pour contourner cet obstacle, nous fixons un paramètre $\beta \in]-1{,}1[$, et nous calculons la confiance comme suit :

- $q(e_i) = (m(e_i) - \beta)/(1-\beta)$ $\forall e_i$ satisfaisant $\beta < m(e_i) \leq 1$.
  e_i aura donc une confiance positive si et seulement si $m(e_i) > \beta$.
- $q(e_i) = (m(e_i) - \beta)/(1+\beta)$ $\forall e_i$ satisfaisant $-1 \leq m(e_i) < \beta$,
  et donc e_i recevra une marge négative si $m(e_i) < \beta$.

En d'autres termes, β permet de créer un décalage entre l'étiquetage du graphe de voisinage d'une part, et la réponse que devrait donner l'oracle d'autre part. Dans le cas où $\beta > 0$, les confiances élevées seront assignées uniquement aux exemples ayant une large majorité d'exemples de leur classe dans leur voisinage, tandis que les exemples ayant une marge faiblement positive se verront attribuer une confiance négative. Dans le cas d'un $\beta < 0$, les exemples incorrectement étiquetés, avec une marge seulement faiblement négative, se verront assigner une confiance positive, tandis que seuls les exemples ayant une large majorité d'exemples de classe opposée dans leur voisinage se verront associer une confiance négative.

Notons que la simulation de l'oracle, telle que nous venons de la présenter, est désormais aisément applicable au cas de données numériques. Cela fera l'objet de la section suivante. Le problème est plus complexe dès lors qu'on exploite des données symboliques. Quel apprenant faible pouvons-nous utiliser ? Quelle distance exploiter pour simuler l'oracle ? La Section 4.3 aura pour objectif de répondre à ces questions en présentant une adaptation de notre modèle à l'inférence grammaticale.

4.2 Résultats expérimentaux sur données numériques

Nous présentons dans cette section les résultats de nos expérimentations, dont le but est d'estimer l'intérêt de notre approche dans le cadre de l'utilisation de données numériques. Nous nous sommes attachés non seulement à estimer le gain de performance par rapport à un apprenant faible boosté avec AdaBoost dans un contexte bruité, mais également à évaluer la vitesse de convergence de notre méthode vers un taux d'erreur optimal, et à comparer celle-ci avec celle de BrownBoost (Freund, 1999), qui reste une des références en matière de gestion du bruit. Le rôle de l'apprenant faible est ici joué par un algorithme de construction de stumps. Dans un premier temps, afin de donner une première intuition sur les comportements comparés de chacune des trois méthodes (AdaBoost, BrownBoost, OraBoost), nous montrons les résultats obtenus par celles-ci

sur une base artificielle. Nous étendons ensuite ces expérimentations à un large éventail de bases de données de l'UCI, auxquelles sont artificiellement appliqués différents niveaux de bruit.

4.2.1 Expérimentations sur une base artificielle

Nous présentons dans un premier temps les expérimentations menées afin d'obtenir une première intuition des performances de OraBoost. Pour pouvoir contrôler au mieux le niveau de bruit présent dans les données d'apprentissage, nous avons choisi de construire une base artificielle, sur laquelle les trois méthodes seront testées. Il s'agit d'une base constituée de trois attributs numériques, et contenant 1000 exemples répartis en deux classes. Pour chacune de ces deux classes, chaque attribut suit une loi de distribution normale. Les paramètres de ces lois ont été sélectionnés de telle manière que le taux de recouvrement entre les deux classes soit important. Ceci nous permettra d'évaluer les capacités de notre algorithme à converger plus vite que les autres face à une erreur bayésienne importante. Ensuite, nous avons bruité cette base en permutant la classe de 20% des exemples tirés aléatoirement. Nous avons comparé les trois méthodes de boosting (AdaBoost, BrownBoost, OraBoost) sur cette base en utilisant une procédure de validation croisée en 10 parties, et des stumps comme apprenants faibles. Notons que nous avons ici utilisé un graphe de 15-ppv pour simuler l'oracle. Les résultats sont présentés dans la Figure 4.1 sous la forme d'une courbe montrant l'évolution des taux de succès de chacune des trois méthodes en fonction du nombre d'itérations.

Au vu des résultats observés, les premières constatations sont les suivantes. Tout d'abord, notons que OraBoost se montre plus performant que AdaBoost en termes de taux de succès. De plus, alors que notre méthode converge rapidement, AdaBoost montre une tendance au sur-apprentissage. En ce qui concerne BrownBoost, si les performances optimales semblent être comparables à long terme avec celles de notre méthode, il apparaît que le temps requis (en termes de nombre d'itérations) pour atteindre ces performances est plus long que dans le cas de OraBoost. Cet exemple montre bien l'intérêt de la prise en compte dans OraBoost, dès le début des itérations, des exemples en région bayésienne. Enfin, il est à noter que BrownBoost se montre être beaucoup moins stable que notre méthode, en présentant une courbe d'évolution assez "accidentée". Ces premières constatations comportementales étant faites sur un exemple jouet, nous allons à présent nous attacher à vérifier leur validité sur une large étude expérimentale impliquant un plus grand nombre de bases de données.

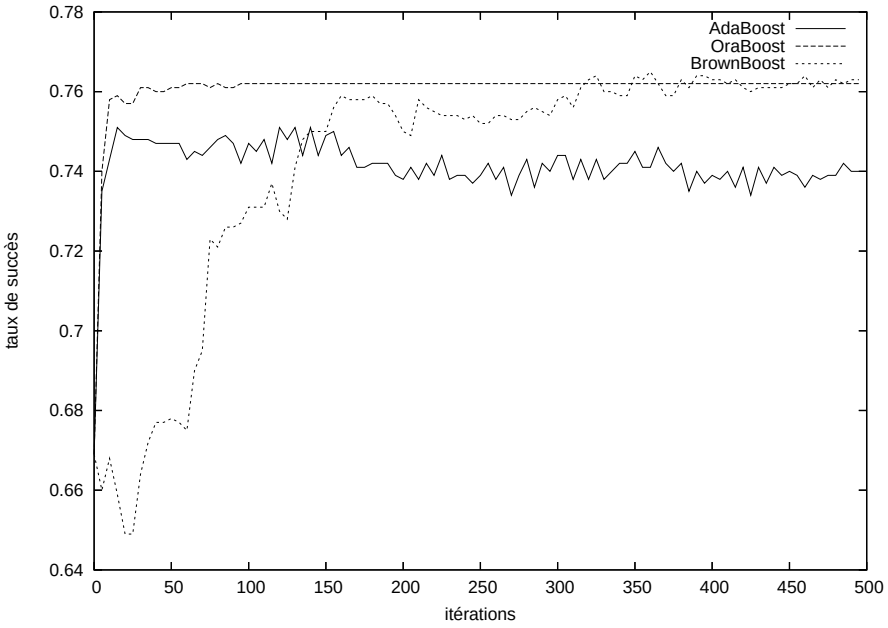


FIG. 4.1 – *Evolution du taux de succès en généralisation de chacune des trois méthodes en fonction des itérations, en utilisant une base de données artificielle.*

4.2.2 Expérimentations sur des données de l'UCI

Le but des expérimentations présentées dans cette section est d'apporter une confirmation statistique des comportements obtenus précédemment.

Protocole expérimental

Nous avons utilisé 10 bases provenant du répertoire UCI[1]. Les caractéristiques de ces bases sont rappelées dans le Tableau 4.1. Nous avons artificiellement ajouté du bruit, de la même manière que celle employée sur la base de données artificielle. Afin d'estimer avec précision l'impact du bruit sur les performances d'ORABOOST, différents niveaux de corruption ont été considérés. Nous avons successivement confronté notre méthode, ADABOOST et BROWNBOOST à 5%, 10%, 15%, 20%, 25% et 30% de bruit. Les résultats des performances atteintes par les trois méthodes au bout de 1000 itérations sur chacune de ces dix bases, et pour les six niveaux de bruit considérés, sont résumés dans les Tableaux 4.2, 4.3 et 4.4. Une moyenne de ces performances au bout de 10, 100 et 1000 itérations est résumée pour chacune des méthodes et pour tous les niveaux de bruit considérés dans le Tableau 4.5. Notons que pour ORABOOST, nous avons testé

[1] http://www.ics.uci.edu/∼mlearn

Base	#exemples	#attributs
Austral	2756	14
Breast	2792	9
Car	3996	6
EchoCardio	521	9
Glass	648	9
Heart	274	13
Pima	3068	11
Vehicle	6762	18
White House	1736	16
Xd6	604	10

Tab. 4.1 – *Caractéristiques des bases de l'UCI employées pour les expérimentations.*

plusieurs valeurs de k dans la simulation de l'oracle, et retenu dans les tableaux celles correspondant aux meilleurs résultats obtenus. Notons enfin que tous ces résultats sont issus d'une validation croisée à 10 parties.

Analyse des résultats

Plusieurs constats peuvent êtres faits des résultats obtenus. D'abord, les premières conclusions tirées sur la base artificielle se trouvent ici confortées. En effet, sans surprise, notre méthode de boosting se montre plus tolérante au bruit qu'AdaBoost. Pour tous les niveaux de bruit, et en moyenne sur les 10 bases, le taux de succès d'OraBoost au bout de 1000 itérations est plus élevé que celui d'AdaBoost. De plus, comme le montre la Figure 4.2, cet écart tend à augmenter avec le niveau de bruit. La comparaison des résultats de OraBoost et BrownBoost permet quant à elle de mettre en avant les très bonnes propriétés de notre méthode en termes de taux de succès, mais aussi de vitesse de convergence. En effet, le Tableau 4.5 nous montre que pour la majorité des niveaux de bruit (4 fois sur 6), OraBoost obtient de meilleurs résultats finaux que BrownBoost. Cette impression est visuellement confirmée par les courbes de la Figure 4.2. De plus, alors que BrownBoost nécessite toujours les 1000 itérations pour converger, OraBoost semble fréquemment atteindre son taux de succès optimal au bout de 100 itérations.

Ces expérimentations confirment donc que nous avons mis en place un modèle permettant de booster efficacement une méthode d'apprentissage utilisant des données numériques bruitées. Si l'application de notre schéma de boosting sur des données numériques a permis d'obtenir de manière assez directe de bonnes performances en simulant l'oracle par un graphe des k-ppv, une réflexion plus

Base	AdaBoost	OraBoost		BrownBoost	AdaBoost	OraBoost		BrownBoost
		k				k		
Breast	91.74	7	93.21	93.94	87.85	7	92.69	93.69
Glass	67.78	17	62.08	67.78	63.19	15	60.69	61.94
Austral	85.29	5	86.32	86.32	86.46	17	86.31	85.87
Heart	70.13	19	70.07	70.72	70.20	13	70.70	70.20
Xd6	71.12	7	71.63	70.87	70.84	15	71.08	70.84
Pima	56.68	7	57.10	56.68	58.70	17	60.87	58.70
White House	92.33	5	95.92	95.29	93.29	5	95.34	94.76
Car	90.57	13	91.51	90.76	87.65	19	91.46	88.02
Vehicle	68.17	11	68.39	67.96	65.88	9	67.16	65.67
Echoc	59.91	15	61.55	59.73	56.82	17	61.45	59.73
Moyenne	75.37	-	75.78	76.00	74.09	-	75.78	74.94

Tab. 4.2 – *Performances sur les 10 bases, en termes de taux de succès en généralisation, de* AdaBoost, OraBoost *et* BrownBoost *après* 1000 *itérations, pour 5% (partie gauche) et 10% (partie droite) de bruit artificiellement ajouté dans les bases.*

Base	AdaBoost	OraBoost		BrownBoost	AdaBoost	OraBoost		BrownBoost
		k				k		
Breast	86.70	7	92.26	94.67	81.73	7	91.23	90.73
Glass	50.54	15	52.86	51.79	55.00	15	53.75	53.75
Austral	84.66	19	84.82	85.27	83.49	7	82.17	85.40
Heart	71.11	13	70.00	70.00	70.85	11	73.24	71.37
Xd6	70.31	21	71.77	70.56	73.73	29	72.76	73.73
Pima	57.27	5	63.58	57.27	61.44	7	61.03	61.01
White House	93.03	9	93.92	94.52	90.70	11	93.10	94.31
Car	84.60	13	86.11	85.75	84.20	25	86.49	84.77
Vehicle	67.41	7	67.83	67.41	64.86	9	68.19	65.07
Echoc	62.36	19	67.27	67.00	55.18	19	69.09	63.64
Moyenne	72.80	-	75.04	74.42	72.12	-	75.10	74.38

Tab. 4.3 – *Performances sur les 10 bases, en termes de taux de succès en généralisation, de* AdaBoost, OraBoost *et* BrownBoost *après* 1000 *itérations, pour 15% (partie gauche) et 20% (partie droite) de bruit artificiellement ajouté dans les bases.*

Base	AdaBoost	OraBoost		BrownBoost	AdaBoost	OraBoost		BrownBoost
		k				k		
Breast	79.91	11	89.12	89.36	81.20	11	89.89	90.15
Glass	50.14	25	58.47	53.36	60.56	21	66.67	60.56
Austral	83.29	9	82.27	85.21	82.49	19	80.86	85.13
Heart	69.44	19	66.67	70.00	68.17	15	73.10	67.67
Xd6	66.17	19	66.63	65.69	62.53	19	63.24	62.77
Pima	56.17	7	60.24	56.17	59.29	11	61.09	58.86
White House	85.86	7	93.23	94.40	82.53	17	95.03	94.45
Car	85.73	25	84.59	85.73	85.31	17	84.75	84.55
Vehicle	67.79	13	68.84	67.72	62.71	15	65.42	63.15
Echoc	59.73	27	65.45	68.09	51.18	27	58.91	56.91
Moyenne	70.42	-	73.55	73.57	69.60	-	73.89	72.42

Tab. 4.4 – *Performances sur les 10 bases, en termes de taux de succès en généralisation, de* AdaBoost, OraBoost *et* BrownBoost *après* 1000 *itérations, pour 25% (partie gauche) et 30% (partie droite) de bruit artificiellement ajouté dans les bases.*

T	AdaBoost	OraBoost	BrownBoost	AdaBoost	OraBoost	BrownBoost
	5%			10%		
10	75.65	74.65	69.70	73.68	75.61	71.43
100	75.39	75.80	73.63	74.08	75.78	72.07
1000	75.37	75.78	76.00	74.09	75.78	74.94
	15%			20%		
10	73.99	74.03	71.97	72.74	74.00	71.39
100	73.56	74.93	73.51	72.99	75.13	71.96
1000	72.80	75.04	74.42	72.12	75.10	74.38
	25%			30%		
10	72.10	72.53	68.95	71.99	73.31	71.11
100	71.58	73.15	71.81	70.70	73.58	71.66
1000	70.42	73.55	73.57	69.60	73.89	72.42

Tab. 4.5 – *Performances moyennes sur les 10 bases, en termes de taux de succès en généralisation, de* AdaBoost, OraBoost *et* BrownBoost *à différentes itérations, pour 5%, 10%, 15% 20% 25% et 30% de bruit artificiellement ajouté dans les bases.*

poussée doit être menée pour adapter cette méthode au cadre des données symboliques. C'est le sujet de la section suivante.

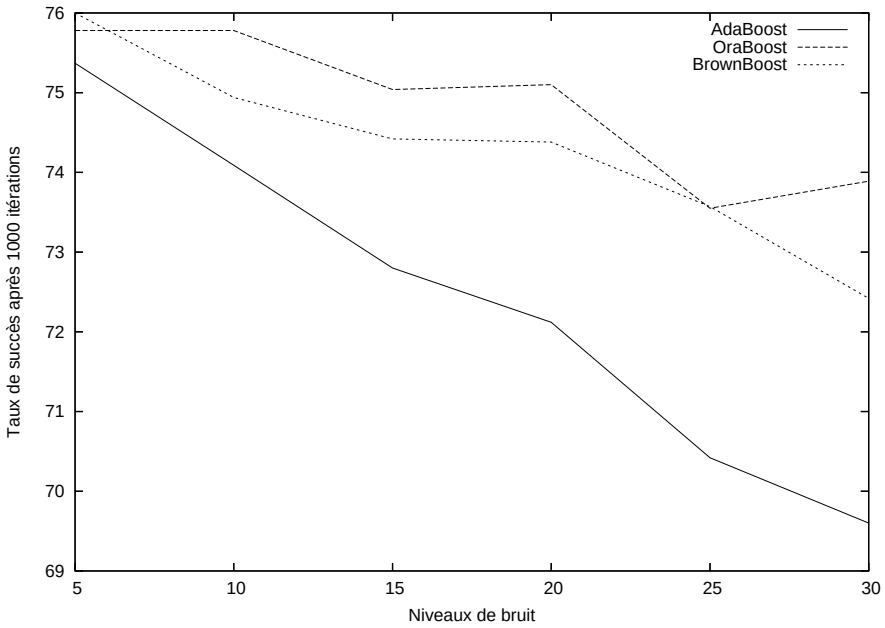


FIG. 4.2 – *Evolution du taux de succès de chacune des trois méthodes après* 1000 *itérations, en fonction du niveau de bruit (moyenne sur les 10 bases).*

4.3 Adaptation aux données symboliques

Pour pouvoir appliquer notre modèle de boosting à des données symboliques, nous devons répondre aux deux questions suivantes : premièrement, quel apprenant faible utiliser, capable de traiter des données structurées, telles que les séquences de symboles ? Deuxièmement, comment simuler notre oracle par un graphe de voisinage sur de telles données ?

Pour répondre à la première question, nous avons opté dans cette section pour des algorithmes d'inférence grammaticale. L'inférence grammaticale est un sous-domaine de l'apprentissage automatique dont le but est d'apprendre des modèles de langages (ensembles de mots). Parmi ceux-ci, les automates finis sont des machines à états qui prennent des mots en entrée et qui les acceptent ou les rejettent. Du point de vue de l'apprentissage automatique, un automate fini est une hypothèse qui sépare l'ensemble de tous les mots en deux classes, la classe dite *positive* des mots acceptés par l'automate, et celle dite *négative* des mots rejetés par celui-ci. Une des raisons expliquant les efforts consentis pour apprendre de

telles hypothèses tient au fait que de nombreux domaines d'application, comme la reconnaissance de la parole, la reconnaissance de formes, le data mining ou le traitement des langues naturelles, tirent partie des propriétés structurelles et sémantiques des automates finis (de la Higuera, 2005 ; Jacquemont et al., 2005). Nous nous intéressons ici aux algorithmes produisant des automates finis par un mécanisme de fusion d'états. On peut répartir de tels algorithmes en deux familles principales. D'une part, les algorithmes d'inférence *exacte*, qui utilisent à la fois des exemples positifs et négatifs du concept à apprendre pour produire un automate fini déterministe (AFD). Citons par exemple RPNI (Oncina & García, 1992), mais aussi EDSM (Lang et al., 1998) ou ESMA (Coste, 1999) qui en sont dérivés. D'autre part, les algorithmes d'inférence *probabiliste*, qui utilisent des exemples positifs uniquement, pour produire des automates finis probabilistes (comme MDI (Thollard et al., 2000), ou encore ALERGIA (Carrasco & Oncina, 1994)). Dans ce qui suit, nous utiliserons RPNI * (Sebban & Janodet, 2003), variante de RPNI, que nous tenterons de booster avec OraBoost.

Pour répondre à la question concernant le graphe de voisinage, nous avons décidé de simuler l'oracle en projetant les mots dans un espace métrique. Nous exploitons la représentation des séquences sous forme de bigrammes pour créer des coordonnées numériques aux mots, et les plonger ainsi dans un espace euclidien. Nous pourrons alors de nouveau faire appel à un graphe des k-PPV.

4.3.1 Quelques bases en inférence grammaticale

RPNI * (Sebban & Janodet, 2003) est une variante de RPNI (Oncina & García, 1992) permettant une gestion de données bruitées durant l'inférence. Avant d'introduire RPNI *, nous présentons RPNI, et mettons en évidence son intolérance aux données bruitées, comme c'est le cas pour l'ensemble des algorithmes d'inférence grammaticale exacte.

Définitions et notations

Un *alphabet* Σ est un ensemble fini non vide de *lettres* et on appelle *mot* sur Σ toute séquence finie $w = a_1 a_2 \dots a_n$ de lettres. On note λ le mot vide. Un *automate fini déterministe* (AFD) est un quadruplet $A = \langle Q,\delta,s_0,F\rangle$ tel que Q soit un ensemble fini d'états, $\delta : Q\times\Sigma \to Q$ soit une fonction de transition, $s_0 \in Q$ soit un état initial et $F \subseteq Q$ soit un ensemble d'états finaux. δ est étendue à l'ensemble de tous les mots en posant $\delta(q,\lambda) = q$ et $\delta(q,xw) = \delta(\delta(q,x),w)$. On dit qu'un mot w est *reconnu* par A si $\delta(s_0,w) \in F$ et qu'il est *rejeté* sinon. Par abus de langage, on dira qu'un état s *contient* un mot w ssi $\delta(s_0,w) = s$. Un mot sera donc accepté par A s'il est contenu par un état final. Ainsi, l'AFD de la Figure 4.3 est défini par $Q = \{0,1\}$, $s_0 = 0$, $F = \{0\}$, $\delta(0,a) = 1$ et $\delta(0,b) = \delta(1,b) = 0$. *abb*

est reconnu car $\delta(0,abb) = \delta(1,bb) = \delta(0,b) = 0 \in F$. Par contre, ba et aa sont rejetés car $\delta(0,ba) = 1 \notin F$ et $\delta(0,aa)$ n'est pas défini.

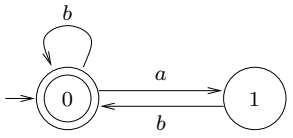


FIG. 4.3 – *Un exemple d'*AFD.

Algorithme 4: *Pseudo-code de* RPNI

Données : Un échantillon d'apprentissage $\langle E_+,E_-\rangle$
Retourner : Un AFD A
$A \longleftarrow$ PTA(E_+); $N \longleftarrow$ Nb d'états de A; $\pi \longleftarrow \{\{q\} : q \in Q\}$;
pour $i = 1$ **jusqu'à** $N - 1$ **faire**
  si $i = \texttt{min}(\pi(i))$ **alors**
    $j \longleftarrow 0$; `continuer` $\longleftarrow$ `vrai`;
    tant que $(j < i)$ **et** `continuer` **faire**
      si $j = \texttt{min}(\pi(j))$ **alors**
        $\pi' \longleftarrow$ `calcule_fusion`(π,i,j,δ);
        si `compatible`$(A/\pi',E_-)$ **alors**
          $\pi \longleftarrow \pi'$; `continuer` $\longleftarrow$ `faux`
        fin
      fin
      $j \longleftarrow j + 1$
    fin
  fin
fin
retourner(A/π)

Fonctionnement de RPNI

Le principe de fonctionnement de RPNI (Oncina & García, 1992) est le suivant. En entrée, RPNI prend un ensemble E_+ de mots reconnus par un AFD cible inconnu (ou exemples) et un ensemble E_- de mots que cet AFD rejette (contre-exemples). Prenons par exemple $E_+ = \{\lambda,b,ab,abb,bab\}$ et $E_- = \{aa,ba\}$. En sortie, RPNI retourne un AFD qui généralise les données de E_+ tout en rejetant les données de E_-. Sous certaines conditions théoriques, RPNI trouve l'AFD cible qui a produit les données[2]. Pour aboutir à cet AFD final, RPNI entreprend tout d'abord la

[2] RPNI identifie polynomialement par données fixées la classe des langages rationnels représentée par des AFD (Oncina & García, 1992).

construction du PTA (*prefix tree acceptor*) des mots de E_+. Ce PTA est le plus grand AFD reconnaissant uniquement les mots de E_+. Ses états sont numérotés selon l'ordre hiérarchique sur les préfixes de E_+ (Oncina & García, 1992). Dans notre exemple, le PTA des mots de E_+ est l'AFD du haut dans la Figure 4.4.

Une fois le PTA construit, RPNI parcourt ses états dans l'ordre. Le traitement du $i$ème état consiste à essayer de le fusionner avec les états 0, 1, ..., $i-1$, dans l'ordre. Fusionner deux états consiste à les regrouper en un seul dont le numéro est le minimum entre les deux états fusionnés. Cet état est final si au moins un des deux états l'était avant fusion. Les transitions sortantes des deux états sont elles-mêmes fusionnées deux à deux lorsqu'elles sont étiquetées par la même lettre, et dans ce cas, les deux états qu'elles pointent doivent alors être fusionnés de manière récursive. Dans la Figure 4.4, RPNI réalise la fusion des états 1 et 0 du PTA. Ceci génère une boucle étiquetée par a sur l'état 0. Comme 1 et 0 ont tous deux des transitions sortantes d'étiquette b, les états 3 et 2 qu'elles pointent sont fusionnés, ce qui conduit à l'AFD du milieu dans la Figure 4.4.

Une fusion *réussit* si aucun exemple de E_- n'est accepté par l'AFD résultant de cette fusion ; elle *échoue* sinon. Dans notre exemple, la fusion de 1 et 0 échoue puisque $aa \in E_-$ est accepté. Donc RPNI abandonne cette fusion et tente la fusion des états 2 et 0, ce qui conduit à l'AFD du bas dans la Figure 4.4. Celui-ci n'accepte aucun exemple de E_- donc la fusion réussit. RPNI considère alors ce nouvel AFD et tente, avec succès, la fusion des états 3 et 0. Celle-ci conduit à l'AFD de la Figure 4.3, résultat final de RPNI sur les données.

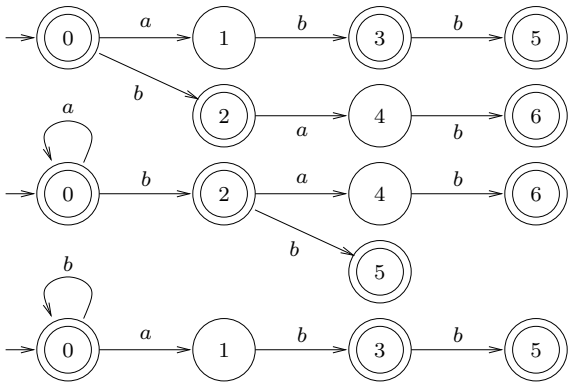


FIG. 4.4 – *L'AFD du haut est le PTA de* $E_+ = \{\lambda,b,ab,abb,bab\}$ *; celui du milieu résulte de la fusion des états 1 et 0 dans le PTA, et celui du bas, de la fusion des états 2 et 0.*

Plus formellement à présent, analysons le pseudo-code de RPNI dans l'algorithme 4. A chaque étape, RPNI maintient une partition déterministe π des états du PTA A de E_+. L'AFD courant, A/π, reconnaît tous les mots de E_+ et rejette

tous les mots de E_-. Par une partition déterministe, nous entendons une partition qui est compatible avec la fonction de transition de A : $\forall p_1,p_2 \in Q,\forall x \in \Sigma,\forall q_1,q_2 \in Q$, si $\pi(p_1) = \pi(p_2)$, $\delta(p_1,x) = q_1$ et $\delta(p_2,x) = q_2$, alors $\pi(q_1) = \pi(q_2)$. $\pi(q)$ désigne l'ensemble des états appartenant au même bloc que l'état q. Fusionner deux états i et j revient à calculer la plus petite partition déterministe π' telle que (1) $\forall s \in Q,\pi'(s) \supseteq \pi(s)$ et (2) $\pi'(i) = \pi'(j)$; elle est obtenue en calculant la fermeture de $\pi \setminus \{\pi(i),\pi(j)\} \cup \{\pi(i) \cup \pi(j)\}$ par la fonction de transition δ du PTA (voir (Dupont & Miclet, 1998) pour plus de détails). Cette fusion réussit si l'AFD A/π' n'accepte aucun mot de E_-. Notons que, par construction, A/π' accepte nécessairement tous les mots de E_+, ce qui préserve l'invariant de la boucle principale. Quant aux conditions techniques $i = \texttt{min}(\pi(i))$ et $j = \texttt{min}(\pi(j))$, elles permettent d'éviter de tenter des fusions qui ont déjà été testées.

4.3.2 Impact du bruit en inférence grammaticale

Supposons maintenant que l'on insère le mot bruité *bbbb* dans l'ensemble E_- de l'exemple précédent. Si l'automate cible était celui de la Figure 4.3, ce mot devrait être positif. Or la Figure 4.5 présente l'automate profondément modifié que RPNI apprend sur ce nouvel échantillon bruité. Cet exemple jouet montre que RPNI n'est pas immunisé contre le sur-apprentissage : la présence de données bruitées induit la construction d'AFD avec un plus grand nombre d'états et de mauvaises performances en généralisation. Pour résoudre ce problème, une approche statistique a été proposée dans (Sebban & Janodet, 2003), traitant le bruit sur les étiquettes, c'est-à-dire un bruit transformant des mots de E_+ en mots de E_-, et réciproquement.

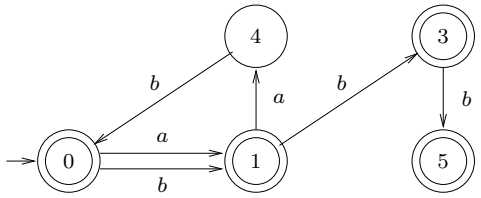


FIG. 4.5 – AFD *inféré par* RPNI *lorsque* $E_+ = \{\lambda,b,ab,abb,bab\}$ *et* $E_- = \{aa,ba,bbbb\}$ *; si l'*AFD *cible est celui de la Figure 4.3, alors bbbb devrait être dans* E_+.

Sebban et Janodet (2003) sont partis du constat suivant : avec RPNI, deux états sont fusionnables si aucun exemple négatif n'est accepté par l'AFD résultant de leur fusion. De manière plus formelle, soit $B = A/\pi = \langle Q,\delta,s_0,F\rangle$ l'AFD résultant de la fusion. Considérons un état s de B et définissons les quantités n_1 et n_2 qui sont respectivement les nombres de mots de E_+ et de E_- contenus par s, *i.e.*,

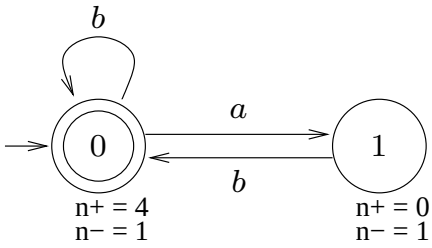


Fig. 4.6 – afd *inféré par* rpni* *; n^+ et n^- représentent respectivement le nombre d'exemples positifs et négatifs contenus dans chaque état.*

$n_1 = |\{w \in E_+ : \delta(s_0,w) = s\}|$ et $n_2 = |\{w \in E_- : \delta(s_0,w) = s\}|$. Dans rpni, s est final dès que $n_1 > 0$. De plus, pour qu'une fusion réussisse, il faut nécessairement que $(n_1 > 0 \Longrightarrow n_2 = 0)$. Mais en présence de bruit, il se peut très bien que s, avant le test de compatibilité, contienne plusieurs mots de E_+ et de E_-, *i.e.* que $n_1 > 0$ et $n_2 > 0$, sans pour autant qu'une erreur d'apprentissage soit en train d'être commise. En effet, prenons l'exemple d'un mot w qui appartiendrait à E_+ du fait du bruit. L'état s qui le contient après fusion acquiert le statut d'état final. Donc tout mot de E_- qui est contenu par s joue le rôle de contre-exemple à la fusion. Cette fusion est donc rejetée par rpni alors qu'elle serait acceptée en l'absence de bruit.

De cette analyse, Sebban et Janodet (2003) ont tiré deux conclusions. D'une part, il faut relâcher la règle d'attribution du statut d'état final à un état, et utiliser désormais la règle de majorité suivante : on dira qu'un état est *positif* (ou final) s'il contient strictement plus de mots positifs (de E_+) que de mots négatifs (de E_-) : $s \in F$ ssi $n_1 > n_2$; on dira qu'il est *négatif* sinon. D'autre part, il faut relâcher la contrainte d'acceptation d'une fusion en permettant à un certain nombre de mots de E_+ et de E_- d'être mal classés par l'afd résultant de la fusion : on dira qu'un exemple négatif (resp. positif) est *mal classé* s'il est contenu par un état positif (resp. négatif) ; de plus, on dira qu'une fusion est *statistiquement acceptable* si la proportion p_2 d'exemples mal classés dans tout l'afd après fusion n'est pas significativement plus grande que la proportion p_1 d'exemples mal classés avant fusion.

Cette manière de faire évite les phénomènes de sur-apprentissage qui sont dus à la présence de bruit. On accepte de réduire, par fusion d'états, la taille des afd appris si elle n'induit pas d'augmentation significative de l'erreur. Ce travail a abouti à l'algorithme rpni* dont le pseudo-code est présenté dans l'Algorithme 5. rpni* fonctionne comme rpni, à ceci près qu'il attribue le statut d'état final aux états contenant une majorité d'exemples de E_+ (avec la fonction `attribue_etat_finaux`), et qu'il utilise un test statistique de comparaison

Algorithme 5: *Pseudo-code de* RPNI*

Données : Un échantillon d'apprentissage $\langle E_+,E_-\rangle$, un paramètre $\alpha \in [0,0.5]$
Retourner : Un AFD
$A \longleftarrow$ PTA(E_+); $B \longleftarrow A$;
$N \longleftarrow$ Nb d'états de A; $\pi \longleftarrow \{\{q\} : q \in Q\}$;
pour $i = 1$ **jusqu'à** $N-1$ **faire**
  si $i =$ `min`$(\pi(i))$ **alors**
    $j \longleftarrow 0$; `continuer` $\longleftarrow$ `vrai`;
    tant que $(j < i)$ **and** `continuer` **faire**
      si $j =$ `min`$(\pi(j))$ **alors**
        $\pi' \longleftarrow$ `calcule_fusion`(π,i,j,δ);
        $B' \longleftarrow$ `attribue_etat_finaux`$(A/\pi',E_+,E_-)$;
        si `statistiquement_compatible`(B,B',E_+,E_-,α) **alors**
          $\pi \longleftarrow \pi'$; $B \longleftarrow B'$; `continuer` $\longleftarrow$ `faux`
        fin
      fin
      $j \longleftarrow j+1$
    fin
  fin
fin
retourner(B)

de proportions pour décider de la réussite ou de l'échec d'une fusion (avec la fonction `statistiquement_compatible`). L'AFD de la Figure 4.6 représente le résultat obtenu par RPNI* sur notre problème bruité précédent. On peut ainsi noter que RPNI* est capable d'inférer l'automate cible.

Dans (Sebban & Janodet, 2003), les auteurs ont montré par leur étude expérimentale que RPNI* améliorait considérablement les performances de RPNI. Cependant, ces bons résultats reposent sur l'hypothèse selon laquelle les exemples mal classés sont réellement bruités. En pratique, cela s'avère souvent exact, d'où l'amélioration des performances montrée par les auteurs. Mais, de par sa nature statistique, RPNI* peut toujours faire une mauvaise fusion, en considérant qu'un exemple mal classé est bruité alors qu'il ne l'est pas.

Pour ces raisons, nous allons désormais tenter d'optimiser les performances de RPNI* à l'aide de notre méthode de boosting. Avant cela, il est nécessaire de construire une simulation de l'oracle dans le cadre de données symboliques.

4.3.3 Simulation d'un oracle à partir de séquences

Comme nous l'avons déjà montré dans le cadre de données numériques, un graphe de voisinage présente des propriétés intéressantes dans l'optique de fournir une estimation de la confiance en l'étiquette d'un exemple. Mais sa construction requiert une distance entre mots. La distance la plus utilisée entre deux séquences x et y est probablement la distance d'édition (Levenshtein, 1965 ; Wagner & Fischer, 1974), qui calcule la suite d'opérations d'édition (insertion, délétion, substitution) la moins coûteuse pour transformer x en y. Quoique simple d'utilisation, cette distance est fortement dépendante des coûts affectés à chaque opération élémentaire. Même si de récents travaux tentent d'apprendre automatiquement le poids de chaque opération (voir (Ristad & Yianilos, 1997 ; Oncina & Sebban, 2006)), il est d'usage de fixer ceux-ci *a priori*, indépendamment du contexte de travail, ce qui limite fortement l'efficacité de la distance d'édition. Pour cette raison, nous avons décidé d'exploiter une distance "orientée données" après une transformation de l'espace de représentation.

Nous avons utilisé des bigrammes comme modèles de langage, afin de décrire les mots sous forme de vecteurs numériques. Les bigrammes sont des modèles de représentation de connaissances qui ont prouvé leur utilité dans le domaine de la reconnaissance de données textuelles. Ils estiment la probabilité d'un mot relativement à un échantillon donné, et permettent donc d'appliquer des méthodes géométriques, ou d'autres méthodes mathématiques, qui sont plus appropriées pour des vecteurs que pour des mots. Nous présentons ci-après le principe de fonctionnement des bigrammes. La probabilité d'une lettre dans un mot dépend seulement de la lettre précédente dans ce mot. Étant donné un mot de n lettres, $w = x_1x_2x_3 \dots x_n$, sa probabilité est :

$$p(w) \quad = \quad \prod_{i=0}^{n} p(x_{i+1}/x_i).$$

Les probabilités élémentaires $p(x_i/x_{i-1})$ sont estimées à partir des mots de l'échantillon d'apprentissage. Enfin, pour donner un sens aux probabilités initiales et finales $p(x_1/x_0)$ et $p(x_{n+1}/x_n)$, deux caractères particuliers, <BOW> (pour Beginning Of the Word) et <EOW> (pour End Of the Word), sont ajoutés à l'alphabet, et l'on pose $x_0 =$<BOW> et $x_{n+1} =$<EOW>.

Afin de permettre une visualisation de nos données, et par là-même d'évaluer graphiquement la qualité de notre oracle simulé, nous avons choisi de construire une représentation géométrique des mots dans un espace euclidien à deux dimensions. Soient $P_+(w)$ et $P_-(w)$ les probabilités de w relativement aux sous-classes positives et négatives de E. Nous estimons ces probabilités par deux bigrammes construits par comptage sur des données positives (pour $P_+(w)$) et négatives (pour $P_-(w)$) d'un échantillon d'exemples. Notons que ces quantités dépendent

fortement de la longueur de w. Aussi, pour permettre la comparaison entre deux mots de longueurs différentes, nous normalisons chacune de ces probabilités en utilisant une moyenne géométrique : $P'_+(w) = \sqrt[n+1]{P_+(w)}$ et $P'_-(w) = \sqrt[n+1]{P_-(w)}$. Maintenant, étant données les valeurs $P'_+(w)$ et $P'_-(w)$, nous pouvons :

1. projeter w en un point de coordonnées $P'_+(w)$ et $P'_-(w)$ dans un espace euclidien à deux dimensions.
2. construire un graphe de voisinage en utilisant la distance euclidienne usuelle dans l'espace de projection.

La Figure 4.7 présente un exemple de la projection pour une base de prénoms français dans laquelle les exemples positifs sont les prénoms féminins, et les exemples négatifs sont les prénoms masculins. Il est à noter ici que ce problème de prénoms français est difficile à apprendre par les algorithmes d'inférence grammaticale exacte. Les performances de base de RPNI ou RPNI* sont en effet très faibles. Or on constate que la projection des exemples dans $\mathbb{R}^2$ permet manifestement de séparer correctement les deux classes. Ceci laisse présager de bonnes performances en tant qu'oracle pour un graphe de voisinage construit dans cet espace. Par ailleurs, comme nous l'avons déjà mentionné, le fait d'utiliser un espace à deux dimensions nous permet d'estimer visuellement la qualité de la projection. Remarquons qu'on pourrait projeter les exemples dans un espace de dimension plus élevée. Il faudrait pour cela utiliser comme attributs des informations numériques permettant de discriminer encore plus les exemples. Par exemple, on pourrait utiliser plusieurs n-grammes avec différentes valeurs de n pour chaque classe.

La Figure 4.8 montre une partie du graphe des k-PPV, avec $k = 7$ construit à partir de la projection de la base de prénoms français présentée précédemment. Ce graphe présente effectivement des regroupements entre prénoms du même genre, et des éloignements entre prénoms de genre opposé. Rappelons à nouveau ici que ce sont les capacités du graphe à fournir des informations sur la confiance en l'étiquette des exemples que nous souhaitons exploiter, et non pas les capacités de la méthodes des k-PPV à correctement classer les exemples de ce problème symbolique. Néanmoins, force est de constater que des recherches futures dans cette direction semblent prometteuses, ce que tend à confirmer Dupont (2006).

4.3.4 Expérimentations et résultats

Protocole expérimental

Nos expérimentations ont été guidées par la nécessité de valider nos résultats théoriques, ainsi que d'estimer les performances d'un graphe de voisinage en tant qu'oracle. Pour effectuer cette tâche, nous avons utilisé trois sortes de bases de données. Les premières sont des benchmarks connus du répertoire UCI : AGARICUS

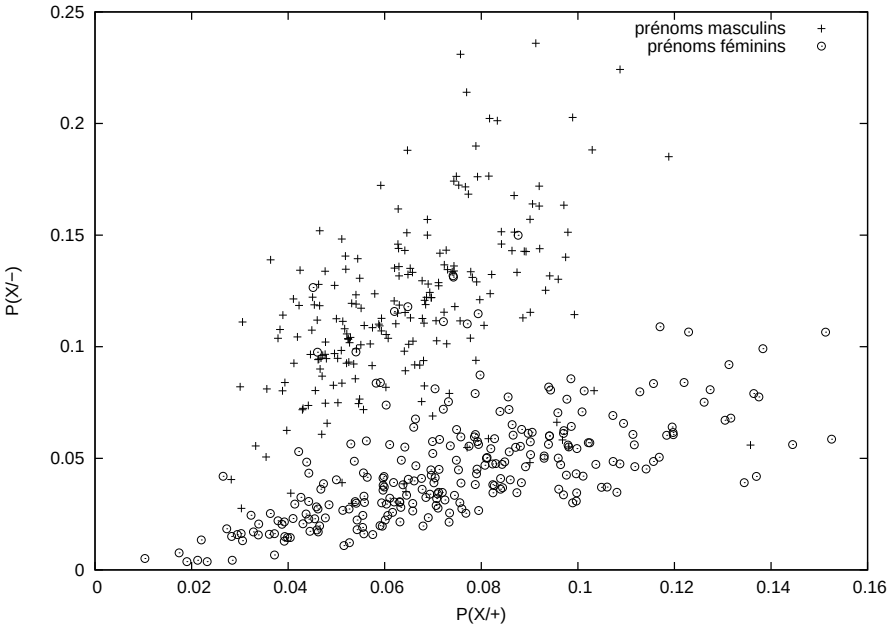


Fig. 4.7 – *Un exemple de projection d'une base de prénoms dans un espace euclidien à deux dimensions à l'aide de bigrammes.*

et TicTacToe. Les secondes sont des bases de données réelles. L'une, appelée USF, provient des Etats-Unis, et recense les mille prénoms les plus fréquemment donnés aux USA en 2002. L'autre, appelée WF, contient tous les prénoms mondiaux commençant par la lettre "a". Pour ces deux bases de données, le concept à apprendre est le sexe correspondant au prénom.

La dernière sorte est constituée de bases artificielles. La taille de l'alphabet utilisé pour les générer varie de quatre à huit lettres selon la base. Deux méthodes de construction sont utilisées. Il s'agit pour chacune d'elle de générer aléatoirement des exemples dont la longueur est comprise dans un intervalle défini, et d'étiqueter comme positifs ceux comportant des *motifs* (*i.e.* des séquences de lettres) fixés. La taille de ces motifs varie de 4 à 6 caractères d'une base à l'autre. La première des deux techniques utilise ensuite d'autres motifs pour étiqueter comme négatifs les exemples qui les contiennent, tandis que la deuxième technique étiquette comme négatifs les exemples ne contenant pas les motifs utilisés pour les exemples positifs. Nous avons généré sept bases de tailles variables ; les six premières contiennent 3 000 mots environ, tandis que la dernière, destinée à évaluer l'intérêt de notre méthode sur un plus grand volume de données, contient 10 000 exemples.

Pour des raisons de temps de calculs importants, liés à la nature même du boosting, nous n'avons pas effectué ici de procédure de validation croisée. Pour

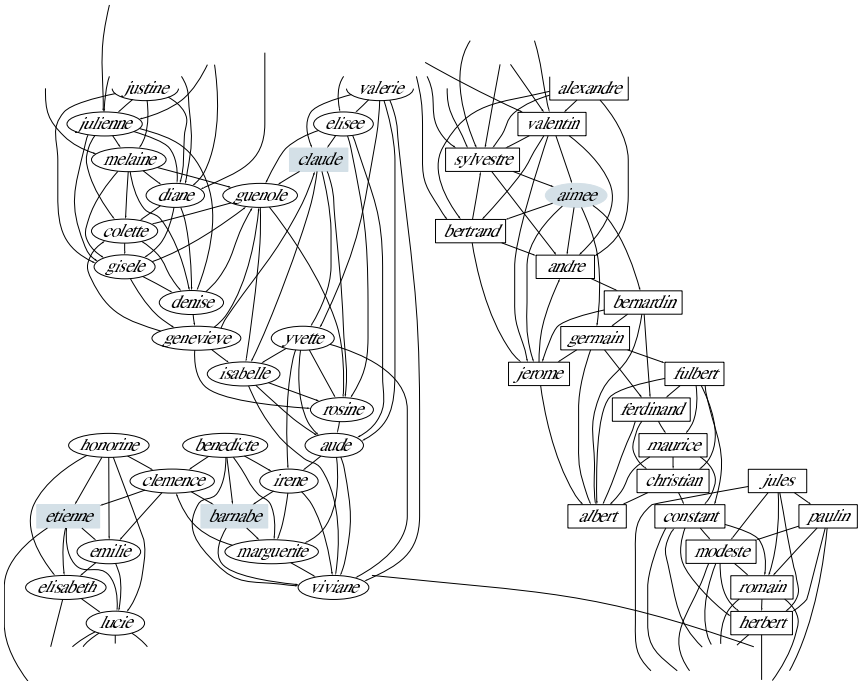


FIG. 4.8 – *Une partie du graphe de voisinage construit sur la projection de la base de prénoms français.*

chacune des onze bases, les exemples ont été répartis aléatoirement en cinq ensembles de taille égale. Quatre de ces ensembles ont été regroupés en un échantillon d'apprentissage, tandis que le dernier a servi d'échantillon de validation.

Enfin, nous avons artificiellement introduit du bruit dans l'ensemble des onze échantillons d'apprentissage, en retournant l'étiquette d'un certain pourcentage de mots. L'estimation des performances en généralisation se fait sur l'échantillon de validation non bruité.

Notons ici, qu'à l'inverse de ce que nous avons fait avec les stumps sur les données numériques, nous n'allons pas appliquer ADABOOST sur RPNI et RPNI*. La raison est simple : RPNI apprenant "par coeur" les données, il ne fait donc pas d'erreur d'apprentissage et ne peut donc pas être boosté. Quand à RPNI*, c'est pour améliorer RPNI qu'il accepte statistiquement d'éventuelles erreurs. Le booster avec ADABOOST reviendrait à confirmer la classe de chaque exemple d'apprentissage et aboutirait aux mêmes résultats que RPNI. Par contre, et c'est ce que nous allons montrer dans cette étude, l'application d'ORABOOST devrait permettre d'améliorer considérablement les performances de RPNI*.

Analyse des résultats

Dans les Tableaux 4.6 et 4.7, nous présentons les résultats que nous obtenons sur ces bases avec rpni, rpni* et OraBoost appliqué à rpni* après 200 itérations. Cette étude expérimentale a été effectuée pour 5, 10, 15 et 20 % de bruit, et en utilisant un graphe des k-ppv construit dans un espace euclidien à deux dimensions, tel que décrit en Section 4.3.3. Le paramètre k a été déterminé en tes-

		5 %				
Base	Taille	shift	rpni	rpni*	OraBoost	perf
Agaricus	5 644	5,0 %	7,2 %	7,2 %	4,3 %	0,0 %
TicTacToe	809	38,9 %	39,5 %	34,5 %	28,3 %	4,9
USF	1 871	32,3 %	33,6 %	31,2 %	26,1 %	26,4
WF	1 887	30,9 %	29,3 %	25,3 %	21,4 %	20,6
Base1	2 540	42,3 %	46,4 %	41,5 %	41,1 %	35,6
Base2	1 505	21,6 %	48,9 %	21,9 %	19,9 %	14,3
Base3	1 532	27,0 %	43,9 %	27,0 %	26,7 %	13,0
Base4	2 969	1,0 %	39,7 %	2,0 %	10,0 %	0,0
Base5	2 179	36,5 %	36,7 %	36,7 %	22,2 %	19,2
Base6	2 004	7,7 %	42,9 %	7,0 %	13,0 %	2,0
Base7	10 000	46,6 %	46,1 %	39,7 %	37,2 %	39,7
Moyenne	2 994,5	26,3 %	37,7 %	24,9 %	22,7 %	16,0
		10%				
Base	Taille	shift	rpni	rpni*	OraBoost	perf
Agaricus	5644	6,2 %	14,0 %	10,0 %	6,9 %	0,0 %
TicTacToe	809	42,6 %	41,9 %	40,7 %	34,5 %	10,5 %
USF	1 871	30,0 %	35,7 %	33,0 %	31,2 %	31,7 %
WF	1 887	34,8 %	29,8 %	31,2 %	22,7 %	26,1 %
Base1	2 540	43,9 %	52,1 %	44,8 %	43,7 %	36,2 %
Base2	1 505	13,6 %	39,2 %	13,6 %	30,5 %	12,3 %
Base3	1 532	39,0 %	39,0 %	39,0 %	12,7 %	12,7 %
Base4	2 969	23,9 %	45,7 %	29,2 %	23,4 %	0,0 %
Base5	2 179	18,6 %	45,9 %	18,6 %	25,0 %	16,2 %
Base6	2 004	42,9 %	45,9 %	42,9 %	25,6 %	1,2 %
Base7	10 000	40,3 %	49,4 %	42,7 %	43,7 %	39,4 %
Moyenne	2 994,5	30,5 %	39,9 %	31,4 %	27,3 %	16,9 %

Tab. 4.6 – *Taux d'erreurs obtenus sur 11 bases avec 5, et 10 % de bruit.*

tant plusieurs valeurs comprises dans [0,15]. Le paramètre β (introduit en Section 4.1), quant à lui, n'a pas fait l'objet d'expérimentations spécifiques durant cette étude. Il a été fixé à 0,5 durant toute l'étude expérimentale, de manière à garantir des valeurs de confiance faibles pour les exemples douteux, même si par ce choix

nous courons le risque de perdre des informations apportées par l'étiquette de certains exemples de marge faiblement positive.

		15 %				
Base	Taille	shift	rpni	rpni*	OraBoost	perf
Agaricus	5 644	9,9 %	14,0 %	10,0 %	6,1 %	0,0
TicTacToe	809	40,7 %	44,0 %	40,7 %	40,7 %	9,8
USF	1 871	32,3 %	34,9 %	32,0 %	27,7 %	27,7
WF	1 887	33,3 %	32,5 %	36,2 %	24,6 %	24,6
Base1	2 540	40,6 %	47,0 %	42,9 %	42,3 %	38,5
Base2	1 505	39,7 %	39,8 %	39,8 %	37,5 %	10,6
Base3	1 532	41,0 %	47,0 %	39,4 %	35,5 %	14,9
Base4	2 969	17,2 %	44,2 %	16,0 %	26,9 %	0,0
Base5	2 179	41,7 %	44,5 %	43,8 %	31,4 %	15,1
Base6	2 004	43,6 %	50,6 %	42,9 %	33,1 %	1,7
Base7	10 000	43,9 %	48,0 %	40,7 %	43,2 %	40,7
Moyenne	2 994,5	34,9 %	40,6 %	34,9 %	31,7 %	16,7
		20 %				
Base	Taille	shift	rpni	rpni*	OraBoost	perf
Agaricus	5 644	13,3 %	23,2 %	18,2 %	11,9 %	0,0 %
TicTacToe	809	41,9 %	40,1 %	40,1 %	38,8 %	11,7 %
USF	1 871	34,4 %	48,8 %	32,2 %	30,6 %	28,8 %
WF	1 887	34,6 %	35,4 %	32,0 %	31,7 %	22,7 %
Base1	2 540	42,9 %	51,2 %	45,7 %	45,2 %	35,2 %
Base2	1 505	44,9 %	46,8 %	46,8 %	45,1 %	12,6 %
Base3	1 532	49,5 %	49,5 %	49,5 %	41,6 %	20,5 %
Base4	2 969	25,4 %	43,4 %	35,0 %	30,4 %	0,0 %
Base5	2 179	42,4 %	41,0 %	39,4 %	34,6 %	9,4 %
Base6	2 004	44,1 %	46,6 %	41,9 %	39,2 %	1,0 %
Base7	10 000	45,4 %	48,0 %	44,3 %	41,8 %	40,5 %
Moyenne	2 994,5	38,0 %	43,1 %	38,6 %	35,5 %	16,6 %

Tab. 4.7 – *Taux d'erreurs obtenus sur 11 bases avec 15 et 20 % de bruit.*

En plus des performances de rpni, rpni* et d'OraBoost, deux colonnes supplémentaires ont été ajoutées aux tableaux. D'abord, nous avons voulu analyser l'écart entre OraBoost et rpni*. Plus précisément, nous désirons connaître, dans cet écart, la part due à l'oracle lui-même, et celle issue directement de notre processus de boosting. Pour atteindre cet objectif, nous avons décidé de retourner l'étiquette des exemples d'apprentissage auxquels le graphe de voisinage attribue une marge négative. Puis nous avons exécuté rpni* sur ce jeu de données (colonne shift). Ensuite, dans l'autre colonne ajoutée (perf), nous présentons les

résultats obtenus avec un oracle parfait, *i.e.* un oracle connaissant exactement les exemples non bruités. L'objectif de cette colonne est d'établir le taux d'erreur optimal (mais inaccessible en pratique sans accès à ce niveau d'expertise) de notre méthode pour chacune des bases.

Les remarques suivantes peuvent être tirées des tableaux de résultats. Tout d'abord, l'utilisation de notre graphe de voisinage pour simuler l'oracle est tout à fait justifiée aux vues des résultats obtenus. Alors que RPNI* améliore déjà significativement RPNI (32,5 % contre 40,3 % en moyenne sur tous les niveaux de bruit), notre modèle boosté est encore plus performant : l'utilisation d'un oracle permet de réduire significativement le taux d'erreur de RPNI* (29,3 % contre 32,5 % en moyenne), comme un test de Student apparié permet de le vérifier.

La colonne SHIFT permet de faire un constat intéressant relatif à l'apport de l'oracle seul. Nos résultats montrent que celui-ci n'est pas toujours très efficace pour améliorer, à lui tout seul, les performances de RPNI*, et que seul un couplage avec notre processus de boosting est presque toujours gagnant. Il s'avère même parfois que les résultats se trouvent détériorés (c'est le cas pour 5% de bruit en moyenne). Une explication de ce phénomène tient à la méthode utilisée pour ré-étiqueter les exemples : en effet, nous avons fait le choix de retourner l'étiquette de tous les exemples de marge négative, même si parmi eux, certains ont une marge légèrement négative. Pour ces exemples, l'oracle émet un faible doute, ce qui n'implique pas de manière catégorique que ces exemples soient des exemples bruités. Retourner la classe de ces exemples induit donc un risque élevé de bruiter un exemple qui ne l'était pas, et donc d'ajouter du bruit dans l'échantillon d'apprentissage (ce qui n'est pas le cas pour les exemples de marge fortement négative), et de ce fait de dégrader les performances de RPNI* lorsque celui-ci apprend sur cet échantillon. En conséquence, nous déduisons de cette colonne SHIFT que la qualité des résultats observés dans la colonne OraBoost est essentiellement liée au schéma même de repondération du boosting, qui utilise l'oracle, plutôt qu'aux seules performances de cet oracle. Enfin, l'ensemble des comportements observés précédemment restent relativement constants lorsque le niveau de bruit augmente.

Pour montrer la convergence exponentielle du taux d'erreur en généralisation au cours des itérations, la Figure 4.9 présente le comportement de l'algorithme sur quatre bases caractéristiques (pour 5 % de bruit). Elle met en évidence l'effet rapide d'OraBoost sur les performances (souvent avant la vingtième itération).

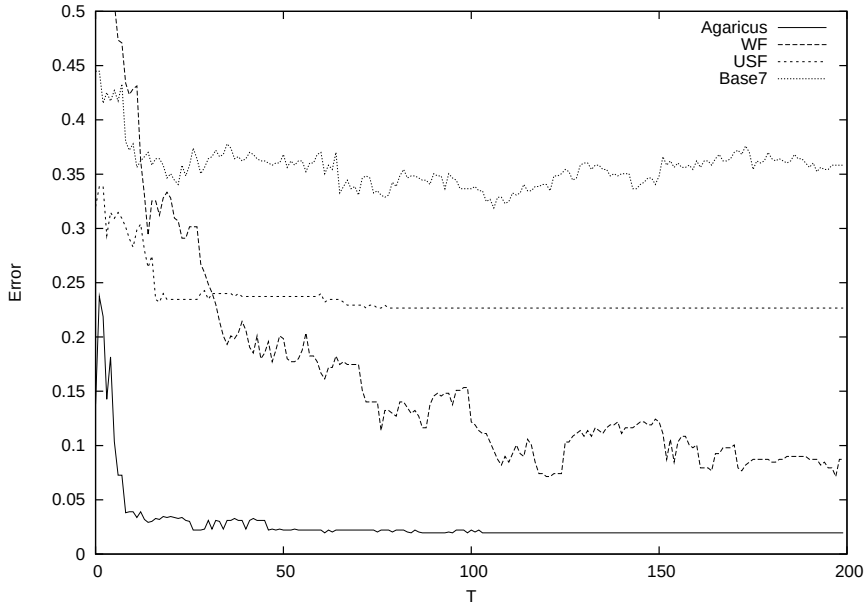


FIG. 4.9 – *L'erreur en généralisation en fonction des itérations*

4.4 Conclusion

Nous avons présenté dans ce chapitre une mise en pratique d'ORABOOST. Celle-ci a nécessité la simulation de l'oracle par un graphe de voisinage construit sur les exemples d'apprentissage. Alors que la génération du graphe des k-PPV est aisée sur des données numériques, nous avons pu montrer qu'il est possible de l'adapter aux séquences de lettres lorsque ces dernières sont projetées dans un espace euclidien avec pour coordonnées leurs probabilités calculées à partir de modèles de bigrammes conditionnels aux classes.

Les résultats expérimentaux ont permis de mettre en évidence l'intérêt de notre algorithme ORABOOST. Non seulement nous obtenons des performances en taux de succès considérablement améliorées, mais nous avons pu constater également, en comparant ORABOOST à BROWNBOOST, que la convergence de notre algorithme est beaucoup plus rapide. Cette propriété permet donc de supprimer, ou tout du moins atténuer, la critique récurrente pouvant être faite au boosting, relative à son coût algorithmique important.

Troisième partie

Adaptation du boosting aux données hétérogènes

Les travaux présentés dans cette partie ont fait l'objet de la publication suivante :

- Suchier, H.M., Janodet, J.C., Largeron, C., Sebban, M. (2006). Boosting d'un pool d'apprenants faibles. *Conférence d'Apprentissage, CAp'06, Trégastel* (à paraître).

5 Boosting et données hétérogènes

Résumé : Nous considérons dans cette partie des problèmes d'apprentissage où les données sont présentées à l'aide de caractéristiques fortement hétérogènes, comme par exemple, une base de personnes dont chaque individu est décrit par son nom (une chaîne de caractères), sa photo (une image), un enregistrement de sa voix (du son) et ses mensurations (des réels). Il n'existe aucun algorithme capable d'apprendre en travaillant sur toutes ces caractéristiques à la fois (sauf en utilisant un codage ce qui peut induire une perte d'information), mais nous disposons d'algorithmes performants et spécialisés pour traiter efficacement chaque type de caractéristiques. Nous proposons une nouvelle procéedure de boosting, k-BOOST, permettant à ces algorithmes de collaborer activement pendant la phase d'apprentissage, et ainsi de construire une hypothèse globale très performante. Nous étudions les propriétés théoriques de k-BOOST, puis nous menons des expérimentations prouvant que notre méthode fonctionne significativement mieux que toute autre combinaison d'hypothèses construites sans collaboration.

5.1 Introduction

Comme nous l'avons vu dans la première partie de cette thèse, les travaux sur les méthodes ensemblistes peuvent être divisés en deux catégories. D'une part, les méthodes dites *homogènes* utilisent une seule et même méthode d'apprentissage pour produire un ensemble d'hypothèses à partir de plusieurs distributions de probabilités sur l'échantillon d'apprentissage. Le boosting fait partie de ces méthodes. Une autre approche, adoptée par les méthodes dites *hétérogènes*, consiste à apprendre des hypothèses de natures différentes (par exemple, sous la forme d'arbres, de réseaux de neurones, de graphes de voisinages, etc) à partir d'une même distribution de probabilités, et de combiner cet ensemble d'hypothèses en

une hypothèse globale plus performante. Toutefois, il est à noter dans ce dernier cas que la notion d'hétérogénéité ne caractérise que les modèles construits, et ne se réfère pas aux données elles-mêmes. Que se passe-t-il lorsque la description des exemples d'apprentissage est composée d'attributs fortement hétérogènes, tels que des séquences, des sons, des images, des arbres, ... ? En fait, dans leurs formes originales, la plupart des méthodes ensemblistes deviennent soit inutiles, et c'est le cas par exemple d'AdaBoost dont le cadre ne prend pas en compte de telles situations, ou insuffisantes pour les méthodes combinant des hypothèses construites uniquement sur les sous-ensembles d'attributs qu'elles peuvent gérer.

Toutefois, des exemples simples montrent que de telles situations peuvent arriver. Par exemple, considérons une base de donnée décrivant des personnes à l'aide de trois attributs : leur prénom, leur taille et leur poid. Si le seul but est d'apprendre le concept "sexe d'une personne", il n'est clairement pas suffisant de n'utiliser que le prénom (et de renoncer aux autres attributs) pour résoudre un tel problème, en particulier parce que beaucoup de prénoms, tels que "Dominique" ou "Claude" par exemple, sont partagés par les deux sexes. Mais d'un autre coté, il serait dommage de ne pas utiliser cette information, et de n'apprendre le concept que sur les deux attributs numériques du fait de l'erreur bayésienne liée à cet espace à deux dimensions. Un autre exemple pourrait être une base de données d'un site de vente en ligne tel que `www.ebay.com`, où les articles sont décrits par une image, une annotation textuelle et un prix. La tâche pourrait ici consister à apprendre l'intérêt d'un acheteur pour les articles. Enfin, prenons comme dernier exemple la base de données biomet (Garcia-Salicetti et al., 2003) qui décrit une personne à l'aide de 5 attributs : son visage, sa voix, ses empreintes digitales, la forme de sa main, et une signature numérique. L'objectif étant de prédire si une personne est un imposteur ou non, l'information fournie par chaque attribut est importante.

Ce type de bases de données ont comme caractéristique d'être composées d'attributs hétérogènes, c'est à dire de données textuelles, images, sons ainsi que des données numériques, et toutes ne peuvent pas être prises en compte par un seul et même algorithme d'apprentissage. En effet, dans le cadre de l'apprentissage de données séquentielles d'une part, l'état de l'art est basé sur des méthodes telles que les n-grammes (Goodman, 2001), ou d'autres algorithmes d'inférence grammaticale (de la Higuera, 2005) (voir aussi le Chapitre 4). Mais ces techniques ne peuvent pas être adaptées pour l'apprentissage à partir de données numériques. D'autre part, nombreux sont les algorithmes qui apprennent efficacement à partir de données numériques, mais ceux-ci ne peuvent pas traiter les données séquentielles directement.

Une première solution pourrait consister à convertir toute les données en un type unique, par exemple numérique, et donc de passer par une phase d'encodage. Or même s'il semble possible de convertir une séquence ou une image en un en-

semble d'attributs quantitatifs telles que des données numériques, le risque d'une telle stratégie est de perdre une partie importante de l'information discriminante contenue dans l'attribut. Pour contourner cet inconvénient, une approche alternative pourrait consister à construire une hypothèse (éventuellement boostée) pour chaque type de données, et de les combiner en une hypothèse globale. Cette idée est développée par Cherkauer (1996). Mais l'inconvénient d'une telle approche est le manque d'interaction des méthodes entre elles durant la phase d'apprentissage. Finalement, une dernière solution est envisageable. Elle consiste en l'utilisation de *cascade generalization* (Gama & Brazdil, 2000) (voir aussi Chapitre 2) : le niveau 0 de la cascade serait construit en utilisant un sous ensemble homogène d'attributs et la méthode d'apprentissage associée, puis le niveau 1 serait construit en ajoutant un nouveau sous-ensemble d'attributs avec les résultats de la première hypothèse, etc. Toutefois, une telle solution n'est possible que dans un nombre très limité de cas, pour lesquels il est possible de fournir séquentiellement les réponses du modèle produit au niveau i en tant qu'attribut de la méthode d'apprentissage du niveau $i+1$. De plus, même s'il existe une coopération entre les méthodes d'apprentissage, celle-ci sera de nature unidirectionnelle, les dernières hypothèses produites n'interagissant avec les premières que par le biais de leurs réponses sur les attributs utilisés au niveau i. Dans cette partie, nous souhaitons

non seulement (*i*) préserver les avantages des méthodes ensemblistes, mis en avant dans la première partie de cette thèse, (*ii*) traiter les données hétérogènes, mais également (*iii*) générer une interaction bilatérale entre les méthodes d'apprentissage. A cette fin, nous nous concentrons sur l'adaptation du boosting à un tel contexte. Une première solution pourrait consister à sélectionner un algorithme pertinent pour chaque type d'attribut et à optimiser ses performances à l'aide de AdaBoost. A la fin des procédures de boosting, on combinerait les k hypothèses résultantes en une hypothèse globale. Toutefois, cette idée n'est pas tout à fait satisfaisante. En effet, d'un point de vue théorique, l'optimisation des performances individuelles de chacune des méthodes d'apprentissage ne garantit pas celle de la méthode finale. De plus, en boostant chacun des apprenants faibles, *indépendamment* des autres, le risque principal serait d'aboutir à une situation de sur-apprentissage, ou bien de ralentir la convergence de l'algorithme. En effet, considérons la description d'une personne constituée de son visage et de sa voix. Supposons maintenant qu'une base de données de telles descriptions contienne des jumeaux, dont les visages sont identiques, mais dont les voix diffèrent. Booster un apprenant faible WL_1 sur les visages, et indépendamment un apprenant faible WL_2 sur les voix, résulterait en une augmentation inutile des itérations de boosting sur WL_1, tandis qu'une collaboration entre les deux méthodes "informerait" WL_1 que la discrimination est possible *via* WL_2, et qu'il est donc inutile de tenter de discriminer les deux visages similaires.

Nous pensons qu'une méthode plus satisfaisante consiste à apprendre les k hypothèses en parallèle *à chaque itération* du boosting, et donc à prendre en compte l'intégralité de l'information issue de la production des k hypothèses dans la règle de mise à jour des poids. Cette stratégie nécessite la mise en place d'un nouveau schéma de repondération, et la vérification de la pérennité des propriétés théoriques du boosting dans ce nouveau cadre. Ce sont les buts de cette partie.

Tout d'abord, nous proposons dans la Section 2 un nouvel algorithme de boosting, appelé k-BOOST. Puis le principal apport de cette partie vise la vérification des propriétés théoriques faisant de notre algorithme une méthode de boosting : dans la Section 3, nous abordons le problème de la convergence de l'erreur empirique. Nous abordons la question de l'erreur en généralisation dans la Section 4. Notons que nous nous concentrerons uniquement sur le cas particulier où $k = 2$, pour lequel nous sommes capable de fournir des résultats théoriques exacts. Enfin, dans la Section 5, nous menons une étude expérimentale pour mettre en avant l'intérêt de notre approche, avant de conclure.

5.2 L'algorithme k-BOOST

Soit E un échantillon d'apprentissage constitué de m exemples. Nous supposons que la description, dans l'espace $\mathcal{X}$, de chacun de ceux-ci est composée d'attributs hétérogènes, et $\mathcal{X}$ est donc un produit cartésien $\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_k$. Ainsi, pour le premier exemple cité en introduction de cette partie, E sera un ensemble de personnes décrites par leurs prénoms, leurs poids et leurs tailles, donc $\mathcal{X}_1$ sera un ensemble de chaînes de caractères, tandis que $\mathcal{X}_2$ sera un ensemble de couples de valeurs réelles correspondants aux poids et aux tailles des individus. Dans un tel cas, il n'existe pas d'algorithme d'apprentissage capable de construire une hypothèse à partir de l'ensemble des attributs de $\mathcal{X}$. En revanche, il est possible d'exploiter la totalité de $\mathcal{X}$ en utilisant un ensemble d'algorithmes, chacun utilisant une partie de $\mathcal{X}$. Par exemple, on pourra utiliser un n-gramme pour exploiter l'information contenue dans $\mathcal{X}_1$, et C4.5 (Quinlan, 1993) pour construire un arbre de décision sur $\mathcal{X}_2$). Pour généraliser ce principe, nous supposons disposer d'un ensemble de k algorithmes, nommés WL_1, ..., WL_k, qui seront utilisés sur les sous-ensembles d'attributs correspondants.

A chaque étape de notre algorithme de boosting, appelé k-BOOST (voir pseudo code de l'Algorithme 6), une distribution $\mathbf{w}_t$ est définie sur E. Ensuite, chaque apprenant faible WL_j utilise les attributs qui lui sont assignés (en fait, ceux qu'il est capable de manipuler) pour construire une hypothèse h_{jt}. Ensuite, $h_{1t}, \ldots, h_{kt}$ sont combinés en un vote pondéré de chacune d'elles, dont la réponse est utilisée pour mettre à jour $\mathbf{w}_t$. Au bout de T itérations, l'hypothèse globale résultante H_T est une combinaison des ensembles d'hypothèses construites au long des ité-

Algorithme 6: Pseudo-code de k-BOOST.

pour $i = 1$ **jusqu'à** m **faire** $w_1(x_i) = 1/m$;
pour $t = 1$ **jusqu'à** T **faire**
  pour $j = 1$ **jusqu'à** k **faire** $h_{jt} = \text{WL}_j(\text{LS},\mathbf{w}_t)$;
  soit $Z_t(u_1,\ldots,u_k) = \sum_{i=1}^{m} w_t(x_i) \exp\left(-\sum_{j=1}^{k} u_j y_i h_{jt}(x_i)\right)$;
  calculer $(c_{1t},\ldots,c_{kt}) \in \mathbb{R}^k$ **tel que** $Z_t(c_{1t},\ldots,c_{kt})$ soit minimal;
  soit $Z_t = Z_t(c_{1t},\ldots,c_{kt})$;
  pour $i = 1$ **jusqu'à** m **faire**
    $w_{t+1}(x_i) = w_t(x_i) \exp\left(-\sum_{j=1}^{k} c_{jt} y_i h_{jt}(x_i)\right) / Z_t$;
  fin
fin
retourner H_T tel que $H_T(x) = \text{signe}\left(\sum_{t=1}^{T}\sum_{j=1}^{k} c_{jt} h_{jt}(x)\right)$

rations de k-BOOST. Notons que lorsqu'un seul apprenant faible est utilisé (c'est à dire, lorsque $k = 1$), k-BOOST coïncide exactement avec ADABOOST, de telle sorte que notre algorithme est une extension de ce dernier. De plus, d'un point de vue algorithmique, il est à noter que k-BOOST est parallélisable, chacun des apprenants faibles pouvant construire son hypothèse indépendamment. Ainsi, en utilisant k machines différentes, notre méthode ne prendra que légèrement plus de temps qu'ADABOOST lorsqu'il est utilisé sur le plus lent des apprenants faibles $\text{WL}_1,\ldots,\text{WL}_k$. Ce surplus de temps est pris par la communication entre les apprenants faibles.

5.3 Résultats sur l'erreur empirique de 2-BOOST

L'erreur empirique $\hat{err}(H_T,E)$ est l'erreur de H_T calculée sur E. Nous montrons dans cette section que $\hat{err}(H_T,E)$ est bornée par une quantité qui décroit avec le nombre d'itérations du boosting. Même si certains de ces résultats peuvent être étendus à k-BOOST (k quelconque), nous nous concentrons, par mesure de simplicité, sur le cas $k = 2$.

5.3.1 Conditions de minimisation de l'erreur empirique

Rappelons tout d'abord la définition de l'erreur empirique :

$$\hat{err}(H_T,E) = (1/m)\sum_{i=1}^{m} [\![H_T(x_i) \neq y_i]\!],$$

où $[\![\pi]\!]$ vaut 1 si le prédicat π est vrai, et 0 sinon. En utilisant 2-BOOST, on obtient le résultat suivant :

Lemme 5.1 $\hat{err}(H_T,E) \leq \left(\prod_{t=1}^{T} Z_t\right)$, *avec*

$$Z_t = \sum_{i=1}^{m} w_t(x_i) \exp\left(-c_{1t} y_i h_{1t}(x_i) - c_{2t} y_i h_{2t}(x_i)\right).$$

Preuve : Soit $A_i = -\sum_{t=1}^{T} \left(c_{1t} y_i h_{1t}(x_i) + c_{2t} y_i h_{2t}(x_i)\right)$. En déroulant la règle de mise à jour de 2-BOOST, nous obtenons $w_{T+1}(x_i) = w_1(x_i) \exp(A_i) / \left(\prod_{t=1}^{T} Z_t\right)$, et sommer $w_{T+1}(x_i)$ pour tout $i \in 1..m$ donne $\left(\prod_{t=1}^{T} Z_t\right) = (1/m) \sum_{i=1}^{m} \exp(A_i)$. Par ailleurs, $[\![H_T(x_i) \neq y_i]\!] = 1$ ssi $H_T(x_i) y_i = -1$, c'est à dire, $A_i \geq 0$. En conséquence, $\exp(A_i) \geq [\![H_T(x_i) \neq y_i]\!]$. Nous pouvons donc déduire que $\hat{err}(H_T,E) \leq (1/m) \sum_{i=1}^{m} \exp(A_i) = \left(\prod_{t=1}^{T} Z_t\right)$. □

En conséquence du Lemme 5.1, la minimisation des $Z_1, \ldots, Z_T$ entraîne celle de l'erreur empirique. Comme dans le cas d'ADABOOST, il s'agit donc pour 2-BOOST de calculer les valeurs c_{1t}, c_{2t} minimisant Z_t. Comme Z_t est une fonction convexe de (c_{1t},c_{2t}), il faut résoudre :

$$\left(\frac{\partial Z_t}{\partial c_{1t}}\right) = \left(\frac{\partial Z_t}{\partial c_{2t}}\right) = 0. \tag{5.1}$$

Pour aborder ce problème, décomposons dans un premier temps Z_t en séparant les éléments de la somme en fonction des valeurs positives et négatives de $y_i h_{1t}(x_i)$ et $y_i h_{2t}(x_i)$ (nous avions déjà expliqué ce principe dans la preuve du Théorème 2.2). Définissons donc, pour tout $b_1,b_2 \in \{-,+\}$, les ensembles $E_t(b_1b_2) = \{i \in 1..m : (y_i h_{1t}(x_i) = b_1 1) \wedge (y_i h_{2t}(x_i) = b_2 1)\}$ et leurs poids $W_t(b_1b_2) = \sum_{i \in E_t(b_1b_2)} w_t(x_i)$. Nous obtenons alors

$$\begin{aligned}
Z_t &= W_t(++)e^{-c_{1t}-c_{2t}} + W_t(+-)e^{-c_{1t}+c_{2t}} \\
&+ W_t(-+)e^{c_{1t}-c_{2t}} + W_t(--)e^{c_{1t}+c_{2t}}, && (5.2)\\
(\partial Z_t/\partial c_{1t}) &= -W_t(++)e^{-c_{1t}-c_{2t}} - W_t(+-)e^{-c_{1t}+c_{2t}} \\
&+ W_t(-+)e^{c_{1t}-c_{2t}} + W_t(--)e^{c_{1t}+c_{2t}} = 0, && (5.3)\\
(\partial Z_t/\partial c_{2t}) &= -W_t(++)e^{-c_{1t}-c_{2t}} + W_t(+-)e^{-c_{1t}+c_{2t}} \\
&- W_t(-+)e^{c_{1t}-c_{2t}} + W_t(--)e^{c_{1t}+c_{2t}} = 0. && (5.4)
\end{aligned}$$

Nous additionnons et soustrayons les Equations (5.3) et (5.4) afin de résoudre l'Equation (5.1), ce qui donne $c_{1t}+c_{2t} = \frac{1}{2} \ln\left(\frac{W_t(++)}{W_t(--)}\right)$ et $c_{1t}-c_{2t} = \frac{1}{2} \ln\left(\frac{W_t(+-)}{W_t(-+)}\right)$.

D'où nous déduisons la proposition suivante :

Proposition 1 *L'erreur empirique de 2-*BOOST *est minimale quand* $\forall t \in 1..T$ *:*

$$c_{1t} = \frac{1}{4}\ln\left(\frac{W_t(++)W_t(+-)}{W_t(--)W_t(-+)}\right), c_{2t} = \frac{1}{4}\ln\left(\frac{W_t(++)W_t(-+)}{W_t(--)W_t(+-)}\right). \tag{5.5}$$

De plus, la valeur minimale de Z_t *est :*

$$2\sqrt{W_t(++)W_t(--)} + 2\sqrt{W_t(+-)W_t(-+)}. \tag{5.6}$$

Notons que l'Equation (5.5) a un sens si et seulement si pour tout $b_1,b_2 \in \{-,+\}$, $W_t(b_1b_2) \neq 0$. Nous supposerons dans la suite que cette condition est vérifiée, mais cela pourra s'avérer faux dans la pratique. Dans ce cas, 2-BOOST s'arrêtera et produira l'hypothèse globale H_{t-1}, de la même manière qu'ADABOOST dans le cas où $W_t(+)$ ou $W_t(-)$ sont nuls (Meir & Raetsch, 2003).

5.3.2 Paramètres caractéristiques de 2-BOOST

Nous avons vu dans le Chapitre 2 que l'erreur empirique d'ADABOOST converge exponentiellement vers 0 avec le nombre T d'itérations. La façon habituelle de le démontrer consiste à montrer que chaque Z_t est significativement inférieur à 1 pour tout $t \geq 1$; ceci se fait en introduisant un paramètre caractéristique d'ADABOOST, noté γ_t, et dénommé *seuil* de l'hypothèse h_t (Meir & Raetsch, 2003). Le but de cette section est d'établir les paramètres caractéristiques propres à 2-BOOST.

Soient X_1 et X_2 deux variables aléatoires qui indiquent le fait que les hypothèses h_{1t} et h_{2t} donnent une réponse correcte. X_1 prendra donc deux valeurs, soit $+1$ quand h_{1t} étiquette correctement un exemple (*i.e.*, $y_i h_{1t}(x_i) = +1$), soit -1 quand h_{1t} commet une erreur ($y_i h_{1t}(x_i) = -1$) ; et de même pour X_2 relativement à h_{2t}. Dans ce contexte, l'ensemble de poids $W_t(b_1b_2)$ décrit la distribution jointe de X_1 et X_2, c'est à dire :

$$\forall b_1,b_2 \in \{-,+\}, W_t(b_1b_2) = \mathbb{P}[X_1 = b_1 1 \wedge X_2 = b_2 1].$$

De plus, de l'Equation (5.2), nous tirons :

$$Z_t(c_{1t},c_{2t}) = \mathbb{E}[\exp(-c_{1t}X_1 - c_{2t}X_2)],$$

et donc Z_t est la *transformée de Laplace* du couple de variables aléatoires (X_1,X_2). Pour une telle transformée, on sait que :

$$\frac{\partial^{p+q} Z_t}{\partial^p c_{1t} \partial^q c_{2t}}(0,0) \;=\; (-1)^{p+q}\mathbb{E}[X_1^p X_2^q], \forall p,q \in \mathbb{N}, \tag{5.7}$$

où $\mathbb{E}[X_1^p X_2^q]$ est l'espérance jointe de X_1 et X_2. En d'autres termes, Z_t est une fonction génératrice de l'espérance, qui détermine complètement et uniquement

la distribution de (X_1,X_2). Avec les Equations (5.2) et (5.7), nous obtenons pour tout $p,q \geq 0$:

$$\begin{array}{ll} \mathbb{E}[X_1^{2p}X_2^{2q}] = \mathbb{E}[1] = 1, & \mathbb{E}[X_1^{2p+1}X_2^{2q}] = \mathbb{E}[X_1], \\ \mathbb{E}[X_1^{2p}X_2^{2q+1}] = \mathbb{E}[X_2], & \mathbb{E}[X_1^{2p+1}X_2^{2q+1}] = \mathbb{E}[X_1X_2]. \end{array}$$

En conséquence, Z_t peut être complètement décrit à l'aide de trois paramètres : $\mathbb{E}[X_1]$, $\mathbb{E}[X_2]$ et $\mathbb{E}[X_1X_2]$ (ainsi que $\mathbb{E}[1] = 1$), puisque toute espérance d'ordre supérieure de (X_1,X_2) est égale à l'une de ces valeurs.

En termes de boosting, $\mathbb{E}[X_1]$ et $\mathbb{E}[X_2]$, que nous notons à présent γ_{1t} et γ_{2t}, sont les seuils des hypothèses h_{1t} et h_{2t}. Elles quantifient la pertinence de chacune des hypothèses h_{1t} et h_{2t} quant à la classe des exemples, puisque γ_{1t} et γ_{2t} sont les espérances des réponses correctes de h_{1t} and h_{2t} :

$$\gamma_{1t} = \mathbb{E}[X_1] = \sum_{i=1}^{m} w_t(x_i)y_ih_{1t}(x_i) \text{ et } \gamma_{2t} = \mathbb{E}[X_2] = \sum_{i=1}^{m} w_t(x_i)y_ih_{2t}(x_i). \quad (5.8)$$

En ce qui concerne $\mathbb{E}[X_1X_2]$, nous pouvons l'exprimer à l'aide de quantités plus compréhensibles, que sont la *covariance* δ_t de X_1 et X_2, et le *coefficient de corrélation* ρ_t de X_1 et X_2 :

$$\delta_t = \mathbb{C}\text{ov}[X_1,X_2] = \sum_{i=1}^{m} w_t(x_i)h_{1t}(x_i)h_{2t}(x_i) - \gamma_{1t}\gamma_{2t}, \quad (5.9)$$

$$\rho_t = \frac{\mathbb{C}\text{ov}[X_1,X_2]}{\sqrt{\mathbb{V}\text{ar}[X_1]}\sqrt{\mathbb{V}\text{ar}[X_2]}} = \frac{\delta_t}{\sqrt{1-\gamma_{1t}^2}\sqrt{1-\gamma_{2t}^2}}. \quad (5.10)$$

Puisque les hypothèses h_{1t} et h_{2t} collaborent pour mettre à jour $\mathbf{w}_t$, il n'est pas surprenant d'avoir ρ_t comme un paramètre important de 2-BOOST : il traduit le degré d'*indépendance* entre X_1 et X_2. D'autres mesures d'indépendances peuvent être utilisées, comme par exemple le coefficient de corrélation inter-classes de X_2 par rapport à X_1, ou la distance χ^2 entre X_1 et X_2, mais nous avons pu constater que ces distances étaient liées à ρ_t, du fait que X_1 et X_2 ne peuvent valoir que $+1$ ou -1.

Nous obtenons :

$$\left\{ \begin{array}{lcl} W_t(++) + W_t(+-) + W_t(-+) + W_t(--) &=& 1, \\ W_t(++) + W_t(+-) - W_t(-+) - W_t(--) &=& \gamma_{1t}, \\ W_t(++) - W_t(+-) + W_t(-+) - W_t(--) &=& \gamma_{2t}, \\ W_t(++) - W_t(+-) - W_t(-+) + W_t(--) &=& \delta_t + \gamma_{1t}\gamma_{2t}, \end{array} \right.$$

$$\iff \left\{ \begin{array}{lcl} W_t(++) &=& (\delta_t + (1+\gamma_{1t})(1+\gamma_{2t}))/4, \\ W_t(+-) &=& (-\delta_t + (1+\gamma_{1t})(1-\gamma_{2t}))/4, \\ W_t(-+) &=& (-\delta_t + (1-\gamma_{1t})(1+\gamma_{2t}))/4, \\ W_t(--) &=& (\delta_t + (1-\gamma_{1t})(1-\gamma_{2t}))/4, \end{array} \right. \quad (5.11)$$

avec les Equations (5.2) et (5.7). Finalement, en insérant les Equations (5.11) et (5.10) dans l'Equation (5.6), nous obtenons :

$$\begin{aligned} Z_t &= \frac{1}{2}\sqrt{\delta_t^2 + 2\delta_t(1+\gamma_{1t}\gamma_{2t}) + (1-\gamma_{1t}^2)(1-\gamma_{2t}^2)} \\ &+ \frac{1}{2}\sqrt{\delta_t^2 - 2\delta_t(1-\gamma_{1t}\gamma_{2t}) + (1-\gamma_{1t}^2)(1-\gamma_{2t}^2)}, \\ &\quad \text{où } \delta_t = \rho_t\sqrt{1-\gamma_{1t}^2}\sqrt{1-\gamma_{2t}^2}. \end{aligned} \tag{5.12}$$

5.3.3 Convergence de l'erreur empirique

Le but de cette section est de fournir une borne sur Z_t qui permettra de montrer la convergence exponentielle de l'erreur empirique de 2-BOOST vers 0. Nous établissons tout d'abord une *hypothèse d'apprentissage faible*, ou WLA (pour Weak Learning Assumption (Kearns & Vazirani, 1994 ; Meir & Raetsch, 2003)), c'est à dire les conditions sous lesquelles WL_1 et WL_2 sont des *apprenants faibles* :

Définition 5.1 *Soit* $E = \{(x_1,y_1),\ldots,(x_m,y_m)\}$ *un ensemble d'apprentissage de taille* m. *Une méthode d'apprentissage* WL *est un apprenant faible par rapport à* E *si et seulement si il existe une constante* $\Gamma > 0$ *telle que pour toute distribution* $\boldsymbol{d}$ *sur* E *et toute hypothèse* $h = \text{WL}(E,\boldsymbol{d})$,

$$\sum_{i=1}^{m} d(x_i)y_i h(x_i) \geq \Gamma.$$

Supposer que WL_1 et WL_2 sont deux apprenants faibles implique qu'il existe deux constantes Γ_1,Γ_2 telles que pour tout $t \geq 1$, $\gamma_{1t} \geq \Gamma_1 > 0$ et $\gamma_{2t} \geq \Gamma_2 > 0$.

Etudions à présent les conditions de convergence de l'erreur empirique. Il est possible d'établir une majoration simple de Z_t, sous forme d'une fonction du coefficient de corrélation ρ_t lorsque γ_{1t} et γ_{2t} sont fixés. En effet, fixons les valeurs de γ_{1t} et γ_{2t}, et étudions la forme de Z_t. Sur la Figure 5.1, il est intéressant d'observer que Z_t est significativement plus petit que 1 pour n'importe quelle valeur de ρ_t, ce qui assure la convergence vers 0 de $e\hat{r}r(H_T,E)$. Bien sûr, nous avons testé plusieurs configurations de γ_{1t} et γ_{2t}, et le comportement de Z_t est à chaque fois le même.

Sous la WLA, et à l'aide de Maple 9.5©, nous pouvons confirmer les remarques précédentes de manière formelle. En effet, en partant de l'Equation (5.12), nous obtenons que : (i) lorsque $0 < \gamma_{1t} \leq \gamma_{2t} < 1$, Z_t atteint un maximum, $\sqrt{1-\gamma_{2t}^2}$, en $\rho_t = \frac{\gamma_{1t}}{\gamma_{2t}}\sqrt{\frac{1-\gamma_{2t}^2}{1-\gamma_{1t}^2}}$, et (ii) lorsque $0 < \gamma_{2t} < \gamma_{1t} < 1$, Z_t atteint un maximum, $\sqrt{1-\gamma_{1t}^2}$, en $\rho_t = \frac{\gamma_{2t}}{\gamma_{1t}}\sqrt{\frac{1-\gamma_{1t}^2}{1-\gamma_{2t}^2}}$. En d'autres termes, nous obtenons :

$$Z_t \leq \sqrt{1-\max(\gamma_{1t},\gamma_{2t})^2}. \tag{5.13}$$

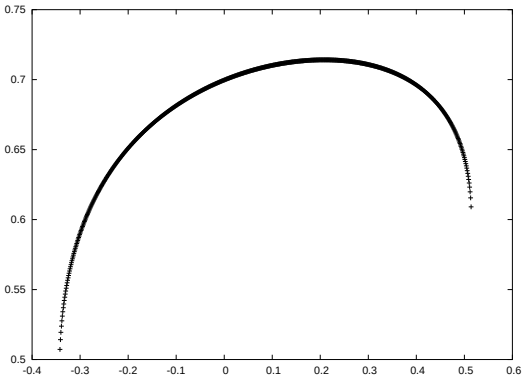


FIG. 5.1 – *Courbe de Z_t en fonction ρ_t, avec $\gamma_{1t} = 0.2$ et $\gamma_{2t} = 0.7$.*

De manière surprenante, ρ_t n'apparaît pas dans cette borne, ce qui signifie que l'erreur empirique de 2-BOOST n'est pas influencée par la corrélation entre h_{1t} et h_{2t}. Cependant, tel ne sera pas le cas pour l'erreur en généralisation. Soit $\Gamma_0 = \max(\Gamma_1,\Gamma_2)$. Nous déduisons :

$$Z_t \leq \sqrt{1-\Gamma_0^2} < \exp\left(-\frac{\Gamma_0^2}{2}\right) < 1.$$

En conséquence, d'après le Lemme 5.1, nous pouvons conclure que :

Proposition 2 *Sous la* WLA, $\hat{err}(H_T,E) < \exp\left(-T\frac{\Gamma_0^2}{2}\right)$, *lorsque* $\Gamma_0 = \max(\Gamma_1,\Gamma_2)$. *L'erreur empirique de 2-*BOOST *converge donc vers* 0 *lorsque* $T \to +\infty$.

De plus, il est à noter que la Définition 5.1 spécifie un apprenant faible par rapport à *toutes* les distributions $\mathbf{d}$ sur E. Or nous devrions nous concentrer uniquement sur les distributions $\mathbf{w}_t$. En fait, cette définition permet de comparer les convergences de ADABOOST et de 2-BOOST : soit ε_{1T} (resp. ε_{2T}) l'erreur empirique de l'hypothèse produite par ADABOOST sur E avec WL_1 (resp. WL_2). On peut montrer que $\varepsilon_{1T} < \exp(-T\Gamma_1^2/2)$ et $\varepsilon_{2T} < \exp(-T\Gamma_2^2/2)$. Comme $\hat{err}(H_T,E) < \exp(-T\Gamma_0^2/2)$ avec $\Gamma_0 = \max(\Gamma_1,\Gamma_2)$, on peut conclure que :

Proposition 3 *La vitesse de convergence de 2-*BOOST *utilisé avec* WL_1 *et* WL_2 *comme apprenants faibles, ne peut pas être plus faible que la plus petite vitesse de convergence de* ADABOOST *utilisé avec* WL_1 *d'une part, et* WL_2 *d'autre part.*

Toutefois, dans la pratique, nous avons pu observer que le comportement de 2-BOOST était souvent plus proche de la moyenne de celui de ADABOOST, utilisé avec WL_1 ou WL_2, que celui du pire de ces deux cas.

5.4 Convergence de l'erreur en généralisation

Dans la pratique, il est fréquent d'observer que l'erreur en généralisation de l'hypothèse finale produite par AdaBoost diminue avec le nombre T d'itérations. Schapire et al. (1997) ont expliqué ce phénomène en établissant un lien entre l'erreur en généralisation et les marges des exemples d'apprentissage (voir Chapitre 2). Récemment, des bornes plus sophistiquées, mais réalistes, ont été proposées dans le but de fournir des explications d'ordre quantitatif (Koltchinskii & Panchenko, 2002). Dans cette section, nous rappelons ces résultats, et les étendons à 2-boost.

5.4.1 Décomposition de l'erreur en généralisation

Soit $\mathcal{H}$ une classe d'hypothèses binaires de $VCdim\ d_{\mathcal{H}}$. Soit co($\mathcal{H}$) l'enveloppe convexe de $\mathcal{H}$, c'est à dire l'ensemble de toute les combinaisons linéaires (finies) d'hypothèses : co($\mathcal{H}$) = $\{f = \sum_i \alpha_i h_i : \forall i, \alpha_i \geq 0$ et $\sum_i \alpha_i = 1\}$. Etant donné un $f \in$ co($\mathcal{H}$) et un exemple x, $f(x) = \sum_i \alpha_i h_i(x)$ est un réel entre $[-1, +1]$; son signe, + ou −, détermine la classe attribuée par f à x ; la *marge* $|f(x)|$ est une mesure de la confiance que f met dans son étiquetage de x.

Etant donné un échantillon $E = \{(x_1,y_1), \ldots, (x_m,y_m)\}$ de m exemples d'apprentissage, tirés indépendamment selon une distribution $\mathcal{D}$ sur $\mathcal{X} \times \{-1,+1\}$, et avec une probabilité d'au moins $1-\delta$, pour tout $f \in$ co($\mathcal{H}$) et $\theta > 0$, Koltchinskii et Panchenko (2002) ont prouvé que l'erreur en généralisation de f est majorée par :

$$\varepsilon^{\theta}(f,E) + \mathcal{O}\left(\frac{1}{\theta}\sqrt{\frac{d_{\mathcal{H}}}{m}}\right) + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{m}}\right). \tag{5.14}$$

Le premier terme de la borne ci-dessus, $\varepsilon^{\theta}(f,E)$, est l'*erreur de marge empirique* de f sur E ; elle traduit la proportion d'exemples d'apprentissage qui sont soit mal classés, soit correctement classés, mais avec une marge θ faiblement positive. Formellement,

$$\varepsilon^{\theta}(f,E) \quad = \quad \frac{1}{m}\sum_{i=1}^{m} [\![y_i f(x_i) \leq \theta]\!].$$

Le reste de l'Inéquation (5.14) est un terme de régularisation. La borne de Koltchinskii et Panchenko (2002) est une amélioration de celle obtenue par Schapire et al. (1997) en supprimant un facteur $\sqrt{\log m}$. Il paraît clair que si f est capable de produire des marges élevées sur E, alors θ et δ peuvent être choisis grands, et donc la partie droite de l'Inégalité (5.14), et l'erreur en généralisation de f sera faible.

5.4.2 Le cas de 2-BOOST

Le résultat ci-dessus est valable pour toute méthode ensembliste, et donc pour 2-BOOST[1] :

$$f_T(x) \quad = \quad \frac{\sum_{t=1}^{T}\left(c_{1t}h_{1t}(x)+c_{2t}h_{2t}(x)\right)}{\sum_{t=1}^{T}\left(c_{1t}+c_{2t}\right)}. \tag{5.15}$$

Toutefois, 2-BOOST a certaines propriétés remarquables. D'une part, l'espace des hypothèses $\mathcal{H}$ utilisé est ici l'union de $\mathcal{H}_1$ et $\mathcal{H}_2$, les espaces respectivement parcourus par WL_1 et WL_2. De par la définition de la $VCdim$, nous pouvons déduire que $d_{\mathcal{H}} = \min(d_{\mathcal{H}_1}, d_{\mathcal{H}_2})$. Donc, à une constante près, le terme de régularisation de l'Inégalité (5.14) est le même que celui de la plus performante des hypothèses construites par ADABOOST avec WL_1 d'un côté, et WL_2 de l'autre.

D'autre part, il est possible de montrer que l'erreur de marge empirique diminue avec le nombre d'itérations. En effet :

Lemme 5.2 $\varepsilon^{\theta}(f_T,E) \leq \left(\prod_{t=1}^{T} Z_{\theta,t}\right)$, *où* $Z_{\theta,t} = Z_t W_t(++)^{\theta/2} W_t(--)^{-\theta/2}$.

Preuve : Soit $A_i = -\sum_{t=1}^{T}\left(c_{1t}y_i h_{1t}(x_i) + c_{2t}y_i h_{2t}(x_i)\right)$ et $B = \theta\sum_{t=1}^{T}(c_{1t}+c_{2t})$. Nous pouvons déduire de l'Equation (5.15) que $[\![y_i f_T(x_i) \leq \theta]\!] = 1$ ssi $A_i + B \geq 0$, ce qui donne $\exp\left(A_i + B\right) \geq [\![y_i f_T(x_i) \leq \theta]\!]$. En conséquence, $\varepsilon^{\theta}(f_T,E) \leq (1/m)\sum_{i=1}^{m}\exp(A_i+B) = \exp(B)\left(\prod_{t=1}^{T} Z_t\right)$, de par la preuve du Lemme 5.1. Enfin, comme $c_{1t} + c_{2t} = (1/2)\ln(W_t(++)/W_t(--))$, nous obtenons $\exp(B) = \left(\prod_{t=1}^{T} W_t(++)^{\theta/2} W_t(--)^{-\theta/2}\right)$, ce qui permet d'obtenir le Lemme. □

Supposons un instant que les hypothèses h_{1t} et h_{2t} soient indépendantes ($\rho_t \simeq 0$). Une telle supposition est souvent formulée dans le but de prouver l'efficacité d'une méthode ensembliste (Dietterich, 2000a). Dans un tel cas, nous déduisons des Equations (5.11) et (5.12) :

$$\left\{\begin{array}{lcl} Z_t & \simeq & \sqrt{(1-\gamma_{1t}^2)(1-\gamma_{2t}^2)}, \\ W_t(++) & \simeq & \frac{(1+\gamma_{1t})(1+\gamma_{2t})}{4}, \\ W_t(--) & \simeq & \frac{(1-\gamma_{1t})(1-\gamma_{2t})}{4}. \end{array}\right.$$

Par le Lemme 5.2, nous obtenons donc :

$$Z_{\theta,t} \simeq (1+\gamma_{1t})^{\frac{1+\theta}{2}}(1-\gamma_{1t})^{\frac{1-\theta}{2}}(1+\gamma_{2t})^{\frac{1+\theta}{2}}(1-\gamma_{2t})^{\frac{1-\theta}{2}}.$$

On peut montrer (Schapire et al., 1997) que si $\theta < \gamma_{1t}/2$, alors $(1+\gamma_{1t})^{\frac{1+\theta}{2}}(1-\gamma_{1t})^{\frac{1-\theta}{2}} < 1$ (de même pour γ_{2t}). Nous pouvons donc conclure :

[1] Puisque $H_T(x) = \text{sign}(f_T(x))$. Notons que l'Equation (5.15) est correcte, c'est à dire que $c_{1t} + c_{2t} > 0, \forall t \geq 1$. En effet, $c_{1t} + c_{2t} = (1/2)\ln(W_t(++)/W_t(--))$ et $W_t(++) - W_t(--) = (\gamma_{1t} + \gamma_{2t})/2$ de par l'Eq.(5.11), donc $W_t(++) > W_t(--) > 0$ sous la WLA.

Proposition 4 *Soit une valeur de marge fixée θ. Si à chaque itération de 2-*BOOST *les hypothèses produites sont (i) indépendantes ($\rho_t \simeq 0$) et (ii) de seuils respectifs γ_{1t} et γ_{2t} supérieurs à 2θ, alors $Z_{\theta,t} < 1$. L'erreur de marge empirique $\varepsilon^{\theta}(f_T,E)$ de 2-*BOOST *converge donc vers 0 avec le nombre d'itérations (par le Lemme 5.2).*

De par l'Inégalité (5.14), l'erreur en généralisation de f_T va alors décroître avec le nombre d'itérations, ce qui sera confirmé de manière expérimentale dans la Section 5.5.

5.4.3 Discussion sur l'hypothèse d'indépendance

En supposant l'indépendance des hypothèses à chaque itération de 2-BOOST, nous avons pu montrer que $Z_{\theta,t} < 1$, ce qui nous a permis de déduire que $\varepsilon^{\theta}(f_T,E)$ converge vers 0. Cette hypothèse d'indépendance peut toutefois être perçue comme étant trop forte d'un point de vue pratique. Néanmoins, nous allons montrer qu'il est possible de s'en passer sans remettre en question la convergence de l'erreur en généralisation. Nous présentons dans la Figure 5.2 la forme de $Z_{\theta,t}$ en fonction du coefficient de corrélation ρ_t, pour des valeurs fixées de γ_{1t}, γ_{2t} et θ. Notons encore une fois que nous avons testé plusieurs valeurs confirmant un comportement similaire à celui observé dans la Figure 5.2.

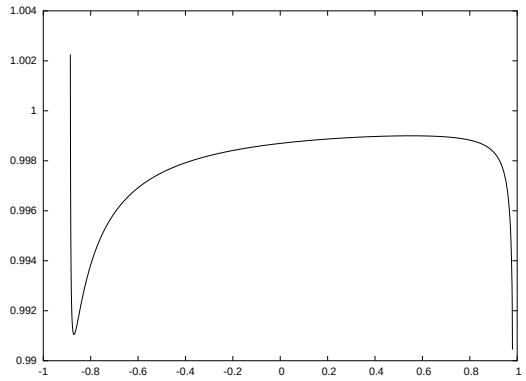


FIG. 5.2 – *$Z_{\theta,t}$ en function de ρ_t avec $\gamma_{1t} = 0.05$, $\gamma_{2t} = 0.07$ et $\theta = 0.02$; $Z_{\theta,t}$ devient infini à partir d'une certaine valeur négative de ρ_t.*

A partir de ce diagramme, les remarques suivantes peuvent être faites :

1. Il paraît clair que lorsque ρ_t est proche de 0, comme nous l'avons supposé dans la Proposition 4, $Z_{\theta,t} < 1$.
2. De plus, nous pouvons remarquer que 2-BOOST aura un bon comportement sur de nouveaux individus si ρ_t est fréquemment fortement positif. En effet, dans un tel cas, h_{1t} et h_{2t} sont d'accord sur les étiquettes de pratiquement

tous les exemples d'apprentissage, et donc ces hypothèses auront probablement le même comportement sur de nouveaux individus. Toutefois, l'intérêt de 2-BOOST est limité dans ce cas, puisque son comportement sera le même que celui de ADABOOST utilisé soit avec WL_1, soit avec WL_2.

3. Finalement, le seul cas qui remet en question notre procédure se produit lorsque ρ_t est fortement négatif. En fait, dans un tel contexte, on peut observer que $Z_{\theta,t} \gg 1$. Ceci n'est pas surprenant, puisque $\rho_t \simeq -1$ signifie que les hypothèses h_{1t} et h_{2t} sont en désaccord sur presque tous les exemples d'apprentissage. Si cela se produit souvent durant la phase d'apprentissage de 2-BOOST, alors l'hypothèse globale f_T, qui résulte de la combinaison de toutes les hypothèses h_{1t} et h_{2t}, se comportera de manière aléatoire sur tout nouvel individu. Toutefois, nous n'avons jamais rencontré en pratique un coefficient de corrélation si fortement négatif entre les hypothèses.

5.5 Résultats expérimentaux

Nous présentons dans cette section les résultats des expérimentations que nous avons menées dans le but d'évaluer les capacités en généralisation de notre méthode. En particulier, nous souhaitons montrer que les performances d'une hypothèse globale produite par 2-BOOST appliqué à deux méthodes d'apprentissage WL_1 etWL_2 sont supérieures en moyenne à celles de n'importe quelle combinaison d'hypothèses fortes produites indépendamment par ADABOOST sur WL_1 et WL_2. A cette fin, nous testerons deux méthodes de combinaison :

Méthode A: Chaque méthode d'apprentissage est boostée individuellement avec ADABOOST; soient

$$f_T(x) = (\sum_{t=1}^{T} c_t h_t(x))/(\sum_{t=1}^{T} c_t)$$

et

$$f'_T(x) = (\sum_{t=1}^{T} c'_t h'_t(x))/(\sum_{t=1}^{T} c'_t)$$

les hypothèses globales résultantes. La méthode A consiste à retourner le signe de $f_T(x) + f'_T(x)$.

Méthode B: Identique à la méthode A, excepté que la combinaison finale consiste à retourner le signe de la somme pondéré $(\sum_{t=1}^{T} c_t) f_T(x) + (\sum_{t=1}^{T} c'_t) f'_T(x)$.

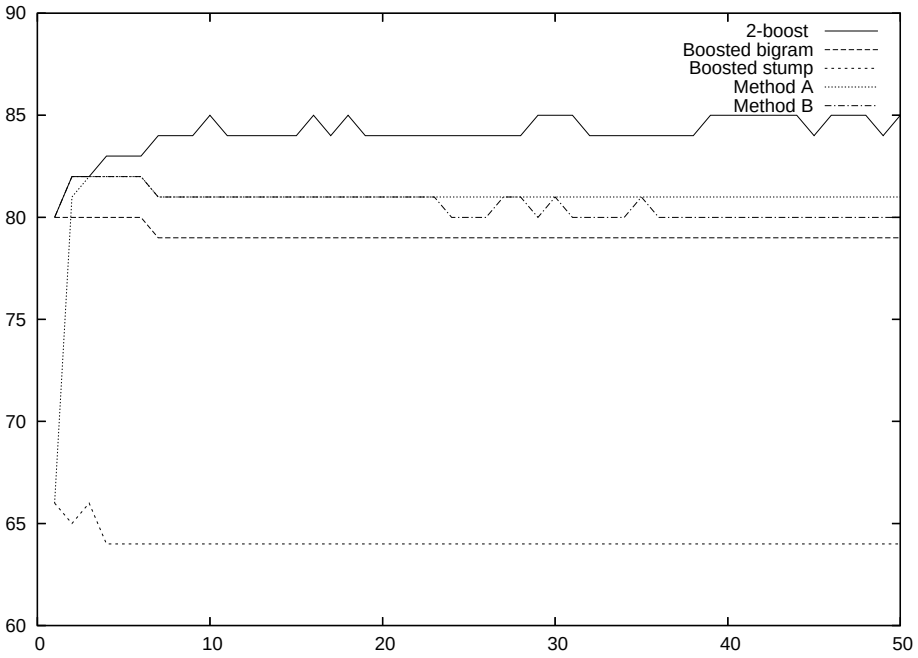
5.5.1 Résultats sur une base simulée

Le but de cette section est de montrer la pertinence de notre approche en présence de données décrites par des attributs fortement hétérogènes. Etant donné le peu de données pré-traitées sur le web, nous avons décidé de construire une base telle que chacun de ses attributs n'apporte pas suffisamment d'information par rapport au concept cible.

Nous avons commencé avec une base contenant 1887 prénoms et le sexe des individus qui peuvent le porter : $+1$ pour un prénom féminin, et -1 pour un prénom masculin. Il est clairement impossible d'apprendre le sexe d'une personne en utilisant uniquement son prénom, à cause du recouvrement des deux classes. Nous avons donc ajouté un nouvel attribut, correspondant à un sport favori, et pouvant être la danse, le tennis ou le football, en supposant que la danse était préférentiellement choisie par les femmes, le football par les hommes, et le tennis indifféremment par les deux. En conséquence, ce nouvel attribut ne permet pas non plus de déduire à lui seul le concept cible.

La tâche consiste à vérifier qu'il est possible de construire une hypothèse de classification prédisant le sexe ($+1$ ou -1) d'une personne en fonction de son prénom et de son sport favori. Nous considérons deux apprenants faibles. L'attribut discret (le sport favori) est géré par un stump. Concernant les prénoms, qui sont des données textuelles, nous avons conçu un apprenant faible basé sur des bi-grammes (Goodman, 2001) (voir aussi Chapitre 4). De façon générale, deux bi-grammes sont construits, un par classe ($+1$ et -1), chacun estimant la probabilité d'une séquence relativement à chaque sexe. L'étiquette de chaque nouvelle séquence est donnée par le bi-gramme maximisant cette probabilité. Bien que le principe soit assez simple, il est à noter qu'il a fallu l'adapter pour prendre en compte une distribution de probabilité $\mathbf{w}_t$ et en faire un apprenant faible.

La Figure 5.3 présente les résultats obtenus (avec une validation croisée à 5 parties) après 50 itérations avec (*i*) 2-BOOST, (*ii*) les deux apprenants faibles boostés individuellement, et (*iii*) leur combinaison par les méthodes A et B. Nous pouvons faire les remarques suivantes. Premièrement, notons que chacune des méthode A et B est plus performante que les deux hypothèses globales correspondant à chacun des apprenants faibles boostés individuellement, non seulement en termes d'erreur en généralisation mais également en termes d'erreur empirique, ce qui signifie que chacun des deux attributs apporte de l'information utile pour l'apprentissage du concept cible. Par ailleurs, 2-BOOST est plus performant que chacune des deux méthodes de combinaison A et B, ce qui prouve l'intérêt de notre procédure de boosting par rapport à la combinaison de deux hypothèses produites par des procédures indépendantes. L'avantage constaté est statistiquement significatif d'après un test de Student apparié.



Méthode	Succès en apprentissage	Succès en généralisation
2-boost	97.73	85.10
Stump boosté	66.11	64.10
Bi-gramme boosté	90.20	79.22
Méthode A	90.26	81.57
Méthode B	92.12	80.87

Fig. 5.3 – *Les courbes représentent l'évolution du taux de succès en généralisation après* 50 *itérations de 2-*boost*, d'un stump boosté, d'un modèle à base de bi-grammes boosté, de la Méthode A, et de la Méthode B. Le tableau montre les résultats moyens après 50 itérations du taux de succès en apprentissage et en généralisation.*

5.5.2 Une comparaison en utilisant des sous-ensembles d'attributs

Dans une seconde série d'expérimentations, nous avons voulu vérifier que le comportement observé dans la section précédente n'était pas lié à la base utilisée. Nous avons donc employé 13 bases provenant de l'UCI. Comme la plupart de celles-ci sont homogènes en termes d'attributs, nous avons simulé l'hétérogénéité en divisant de façon aléatoire l'ensemble des attributs en deux sous ensembles ($\mathcal{X}_1$, $\mathcal{X}_2$) de tailles égales. Nous avons appliqué 2-boost avec deux apprenants faibles : une méthode de construction de stump, d'une part, et une méthode bayésienne naïve d'autre part (John & Langley, 1995). Le Tableau 5.1 (colonne Expé. 5.5.2) présente les résultats obtenus dans cette configuration.

Pour chacune des bases, nous donnons sa taille $|E|$, son nombre initial d'attri-

buts #Attr, et les performances en généralisation (obtenues par validation croisée à 5 parties) de 2-BOOST, de la Méthode A et de la Méthode B. De plus, nous avons souligné le résultat obtenu par la méthode la plus performante. Plusieurs remarques peuvent être faites en observant ce tableau. Premièrement, notre méthode est la plus performante sur 9 des 13 bases, contre 4 pour la méthode B et aucune pour la méthode A. De plus, nous avons calculé les performances moyennes, en pondérant chaque valeur individuelle par la taille de la base à laquelle elle correspond. 2-BOOST atteint un taux de 82.70% ce qui est beaucoup plus que les 75.97% de la Méthode A (+6.73 en faveur de 2-BOOST) et significativement plus que les 81.19% de la Méthode B (+1.51).

En analysant les résultats relatifs aux tailles des échantillons d'apprentissage, nous pouvons également faire la remarque suivante. L'avantage de 2-BOOST comparativement à la Méthode B (qui est la plus proche en termes de performances) semble être plus important en moyenne lorsque les bases de données sont de petites tailles. En effet, les performances moyennes pour les bases contenant moins de 2000 individus est d'environ 77.8% pour 2-BOOST et 75.6% pour la Méthode B (+2.2), tandis que cet écart est de seulement de +1.3 pour les bases de plus de 200 individus. Ce résultat met en évidence la nécessité, lorsque peu d'exemples d'apprentissage sont disponibles, d'une collaboration durant la phase d'apprentissage entre les méthodes d'apprentissage.

			Expé. 5.5.2			Expé. 5.5.3		
Base	$\|E\|$	#Attr	2-BOOST	Méthode A	Méthode B	2-BOOST	Méthode A	Méthode B
Bigpole	1996	5	67.59	62.32	63.48	68.04	67.53	67.48
Horse	1468	23	79.90	73.50	78.68	85.35	76.63	84.60
Austral	2756	15	86.97	73.00	86.39	87.26	87.84	87.45
Balance	2496	5	92.05	71.39	89.51	98.10	97.14	97.46
Breast	2792	10	96.24	95.88	96.67	97.39	96.10	96.45
German	1004	25	73.10	73.30	73.60	73.10	73.30	73.60
Glass	167	10	74.40	72.81	72.61	81.65	79.95	81.03
Heart	274	14	79.19	79.17	79.91	81.02	81.02	78.81
Ionosphere	736	35	98.91	92.67	93.08	92.26	91.03	91.03
Pima	3068	9	73.01	72.62	72.62	73.01	72.62	72.62
TicTacToe	2396	10	78.96	71.62	74.96	91.95	90.19	92.41
WhiteHouse	439	17	96.89	95.80	95.05	98.30	97.12	97.41
xd6	604	11	74.83	70.86	75.33	75.82	75.49	75.49
Moyenne	1728	14	82.70	75.97	81.19	85.60	84.34	85.22

TAB. 5.1 – *Comparaison de 2-*BOOST *avec les Méthodes A et B sur 13 bases. Pour l'Expérimentation 5.5.2, chaque apprenant faible est appliqué à un sous-ensemble des attributs originaux. Pour l'Expérimentation 5.5.3, chaque apprenant faible est appliqué à l'ensemble des attributs originaux.*

5.5.3 Résultats sur l'ensemble des attributs

Dans cette dernière série d'expérimentations, nous avons voulu vérifier si 2-boost restait efficace par rapport aux méthodes A et B, dans le cas de bases de données *homogènes*. En d'autres termes, que se passe-t-il quand l'ensemble des attributs originaux est utilisé par chacune des deux méthodes d'apprentissage? Est-il encore pertinent d'utiliser 2-boost? Le Tableau 5.1 (colonne Expe. 5.5.3) présente les résultats obtenus à l'aide d'une validation croisée à 5 parties.

Tout d'abord, il est à noter que la différence en faveur de notre approche, entre 2-boost et les Méthodes A et B est considérablement réduite. Ce comportement n'est pas surprenant puisque dans ces deux dernières méthodes, les apprenants faibles ont maintenant accès à tous les attributs, soit une plus grande quantité d'information. Malgré cela, il est à noter que la différence entre 2-boost et les Méthodes A et B reste significative au sens d'un test de Student apparié. De plus, ces résultats confirment la pertinence et la stabilité de notre approche puisque nous obtenons les meilleurs résultats sur 10 des 13 bases.

5.6 Conclusion

Pour autant que nous sachions, 2-boost est la première procédure de boosting capable de gérer, par un processus collaboratif, des bases de données contenant des attributs hétérogènes. Nos résultats expérimentaux semblent donc très prometteurs. Toutefois, même si la majorité des résultats théoriques présentés dans cette partie sont assez facilement généralisables à k-boost, certains résultats de convergence semblent difficiles à établir pour $k > 2$. Un premier problème vient du calcul des valeurs optimales de $c_{1t}, \ldots, c_{kt}$ minimisant Z_t, qui peuvent seulement être approximées en utilisant une méthode standard de Newton-Raphson. De même, pour l'interprétation probabiliste de Z_t en termes de transformée de Laplace, il apparaît nécéssaire d'utiliser les espérances pour décrire Z_t. Or le nombre exponentiel de ces paramètres rend les propriétés très difficiles à démontrer. Cependant, en l'état actuel de nos intuitions, nous pensons que l'erreur empirique, comme l'erreur en généralisation devraient décroître avec le nombre d'itérations.

Conclusion Générale

Conclusion Générale

Nous avons choisi délibérément, tout au long de ce manuscrit de thèse, de rédiger une conclusion spécifique à chacun des chapitres présentés. La motivation principale de ce choix est venue de la possibilité offerte dans un tel cas d'y spécifier des aspects parfois localement très techniques liés aux modèles proposés.

Cette conclusion générale a donc pour vocation à prendre du recul sur le travail effectué durant cette thèse, et à ouvrir ainsi la porte à des perspectives de recherche qui nous semblent prometteuses.

La première contribution de cette thèse a consisté à proposer une méthode de boosting tolérante aux données bruitées, nommée OraBoost. Cet algorithme exploite une information complémentaire, fournie par un oracle, sur le niveau de confiance en l'étiquette associée à chaque exemple. En pratique, cet oracle est simulé à l'aide d'un graphe de voisinage construit sur l'échantillon d'apprentissage. Les résultats théoriques présentés ont surtout mis l'accent sur les propriétés de l'erreur empirique en termes de bornes et de convergence. Concernant le comportement théorique d'OraBoost en généralisation, nous n'avons pas fourni de nouveaux résultats par rapport à ceux présentés par Freund et Schapire dans le cadre d'AdaBoost. A la pondération près des exemples, différente dans notre algorithme OraBoost, le schéma de boosting n'ayant pas été remanié, il n'y a en effet aucune raison que les résultats de convergence en généralisation basés sur les marges des exemples soient remis en cause. Tout au moins, nous pouvons penser qu'en exploitant l'information sur les marges des exemples par le biais du graphe des k-ppv, cette convergence puisse être accélérée. Les résultats expérimentaux ont parfaitement confirmé cette intuition.

Néanmoins, concernant ce graphe, il est indéniable que des avancées significatives peuvent être effectuées au niveau de la qualité de l'estimation du voisinage de chaque exemple. Alors que nous avons exploité jusqu'à présent une distance euclidienne sur une projection des mots dans un espace métrique par des modèles de bigrammes, les derniers travaux notamment de (Oncina et Sebban 2006) portant sur l'apprentissage de distances stochastiques sous la forme de transducteurs conditionnels ouvrent des perspectives d'améliorations considérables. En effet, quoi de plus informatif que la probabilité d'un mot conditionnellement à son voisinage pour juger de la vraisemblance (et donc de la confiance) de son étiquette ? De plus, de nouveaux travaux portent actuellement sur l'apprentissage de telles distances sur les données arborescentes. Ainsi, adapter OraBoost à des apprenants faibles tels que des automates d'arbres probabilistes ou des transducteurs d'arbres semble être une perspective extrêmement motivante.

Notre deuxième contribution a porté ensuite sur un nouveau schéma de boosting intégré dans l'algorithme k-BOOST. Nous avons ainsi pu montrer qu'il était possible d'utiliser une méthode de boosting sur des données constituées d'attributs hétérogènes, ces derniers caractérisant de plus en plus les bases de données modernes. Le schéma de boosting classique ayant été assez profondément remanié, nous avons cette fois fourni des bornes théoriques non seulement sur l'erreur empirique mais également sur l'erreur en généralisation. Cependant, ces résultats n'ont pu être obtenus que dans le cas particulier de 2-BOOST. Même si avoir un schéma gérant deux types d'hypothèses différentes est déjà satisfaisant, les résultats théoriques obtenus pour 2-BOOST devront être généralisés à un k quelconque, probablement par le biais de méthodes d'estimation.
D'un point de vue expérimental, cette fois, il est objectif de dire que notre étude concernant 2-BOOST n'a pas été très diversifiée puisque nous avons été contraints soit de générer artificiellement les données, soit de considérer certaines bases homogènes comme étant composées d'attributs de natures différentes. La raison vient probablement du fait que les benchmarks exploités par la communauté en apprentissage automatique, comme le répertoire de l'UCI, ne se sont pas encore mis "au goût du jour", et présentent quasiment toujours des attributs de même type. Il est vrai que dès que des images et du son deviennent accessibles par le biais de bases de données partagées, des problèmes de droits, de préservation de vie privée, etc., émergent rapidement. En attendant donc leur diffusion sur le Net, nous avons commencé à développer un prototype permettant de capturer des photos d'individus (des étudiants parfaitement consentants !), des messages vocaux, ainsi que des phrases manuscrites, pour constituer une base de données hétérogènes, qui nous permettra probablement de mettre encore plus en évidence l'aspect incontournable de modèles comme celui que nous avons développé dans cette thèse.

Enfin, une dernière voie d'investigation intéressante consisterait en la fusion des deux modèles de boosting présentés dans cette thèse. En effet, le cas de figure où des données d'apprentissage seraient non seulement bruitées, mais également constituées d'attributs de natures hétérogènes, est tout à fait envisageable, voire plus que probable dans les années à venir. Il serait alors intéressant de pouvoir bénéficier des avantages offerts par nos deux approches. Si en pratique cette idée paraît séduisante, elle ouvre également la voie à de nombreuses investigations futures, comme l'étude des mesures de similarités entre données structurées, ou encore la vérification des propriétés théoriques des modèles ainsi fusionnés.

Bibliographie

Ali, K. (1995). *Learning probabilistic relational concept descriptions*. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.

Ali, K. M., & Pazzani, M. J. (1995). *On the link between error correlation and error reduction in decision tree ensembles* (Technical Report ICS-TR-95-38). University of California, Irvine.

Ali, K. M., & Pazzani, M. J. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, *24*, 173–202.

Angluin, D. (1987a). Learning regular sets from queries and counterexamples. *Information and Control*, *39*, 337–350.

Angluin, D. (1987b). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Bagui, S. C., & Pal, N. R. (1995). A multistage generalization of the rank nearest neighbor classification rule. *Pattern Recognition Letters*, *16*, 601–614.

Bahler, D., & Navarro, L. (2000). Methods for combining heterogeneous sets of classifiers. *17th National Conference on Artificial Intelligence (AAAI 2000)*.

Berger, A. (1999). Error-correcting output coding for text classification. *IJCAI'99: Workshop on machine learning for information filtering, 1999*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT: Proc. of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1996b). *Bias, variance, and arcing classifiers* (Technical Report 460). Statistics Department, University of California.

Breiman, L. (1998). *Arcing classifiers* (Technical Report 460). Statistics Department, University of California.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Comput.*, *11*, 1493–1517.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Breiman, L., Friedman, J. H., Olshen, R. H., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth, Belmont, CA.

Brodley, C., & Friedl, M. A. (1996). Identifying and eliminating mislabeled training instances. *Proc. of the 13th National Conference on Artificial Intelligence* (pp. 799–805).

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion Journal (Special issue on Diversity in Multiple Classifier Systems)*, *6*, 5–20.

Buntine, W. (1990). *A theory of learning classification rules*. Doctoral dissertation, Universityof Technology, School of Computing Science, Sidney, Australia, Sydney.

Carrasco, R., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. *Proc. 2nd International Colloquium on Grammatical Inference - ICGI '94* (pp. 139–150). Springer-Verlag.

Chan, P. K., & Stolfo, S. J. (1993). Toward parallel and distributed learning by meta-learning. *Working Notes AAAI Work. Knowledge Discovery in Databases* (pp. 227–240).

Cherkauer, K. (1996). Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models* (pp. 15–21).

Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, *10*, 57–78.

Coste, F. (1999). *State merging inference of finite state classifiers* (Technical Report). Publication interne n°1250.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, *IT13*, 21–27.

Dasgupta, S., & Long, P. M. (2003). Boosting with diverse base classifiers. *Proc. of the 16 th International Conference on Computational Learning Theory* (pp. 273–287).

de la Higuera, C. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning*, *27*, 125–138.

de la Higuera, C. (2005). A bibliographic survey on grammatical inference. *Pattern Recognition*, *38*, 1332–1348.

de Oliveira Jr., J. J., Kapp, M. N., de Almendra Freitas, C. O., de Carvalho, J. M., & Sabourin, R. (2004). Handwritten recognition with multiple classifiers for restricted lexicon. *SIBGRAPI* (pp. 82–89).

Dietterich, T. (1997). Machine learning research: for current directions. *AI Magazine*, *18*, 97–136.

Dietterich, T. G. (2000a). Ensemble methods in machine learning. *Lecture Notes in Computer Science*, *1857*, 1–15.

Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*, 139–157.

Dietterich, T. G., & Bakiri, G. (1991). Error-correcting output codes: a general method for improving multiclass inductive learning programs. *Proc. of the Ninth AAAI National Conference on Artificial Intelligence* (pp. 572–577). Menlo Park, CA: AAAI Press.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Dietterich, T. G., & Kong, E. B. (1995). *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms* (Technical Report). Department of Computer Science, Oregon State University.

Domingo, C., & Watanabe, O. (2000). MadaBoost: A modification of AdaBoost. *Proc. 13th Annu. Conference on Comput. Learning Theory* (pp. 180–189). Morgan Kaufmann, San Francisco.

Duffy, N., & Helmbold, D. P. (1999). Potential boosters? In (Solla et al., 2000), 258–264.

Dupont, P. (2006). Noisy sequence classification with smoothed markov chains. *Actes de la huitième Conférence francophone sur l'apprentissage automatique, Trégastel*. à paraître.

Dupont, P., & Miclet, L. (1998). *Inférence grammaticale régulière : fondements théoriques et principaux algorithmes* (Technical Report RR-3449). INRIA, Rennes, France.

Dzeroski, S., & Zenko, B. (2002). Stacking with multi-response model trees. *MCS '02: Proc. of the Third International Workshop on Multiple Classifier Systems* (pp. 201–211). London, UK: Springer-Verlag.

Dzeroski, S., & Zenko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, *54*, 255–273.

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. Chapman and Hall, London.

Efron, B., & Tibshirani, R. (1995). *Cross-validation and the bootstrap: Estimating the error rate of a prediction rule* (Technical Report 477). Dept. of Statistics, Stanford University.

Fan, W., Stolfo, S. J., & Zhang, J. (1999). The application of adaboost for distributed, scalable and on-line learning. *KDD '99: Proc. of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 362–366). New York, NY, USA: ACM Press.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Inf. Comput.*, *121*, 256–285.

Freund, Y. (1999). An adaptive version of the boost by majority algorithm. *COLT '99: Proc. of the twelfth annual conference on Computational learning theory* (pp. 102–113). New York, NY, USA: ACM Press.

Freund, Y., & Schapire, R. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, *5*, 771–880. Appeared in Japanese, translation by Naoki Abe.

Freund, Y., & Schapire, R. E. (1996a). Experiments with a new boosting algorithm. *International Conference on Machine Learning* (pp. 148–156).

Freund, Y., & Schapire, R. E. (1996b). Game theory, on-line prediction and boosting. *COLT '96: Proc. of the ninth annual conference on Computational learning theory* (pp. 325–332). New York, NY, USA: ACM Press.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Friedman, J., Hastie, T., & Tibshirani, R. (1998). *Additive logistic regression: a statistical view of boosting* (Technical Report). Dept. of Statistics, Stanford University Technical Report.

Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning*, *41*, 315–343.

Garcia-Salicetti, S., Beumier, C., Chollet, G., Dorizzi, B., Jardins, J. L.-L., Lunter, J., Ni, Y., & Petrovska-Delacretaz, D. (2003). Biomet: A multimodal person authentication database including face, voice, fingerprint, hand and signature modalities. *Fourth International Conference on Audio and Video-Based Biometric Person Authentication*.

Giacinto, G., & Roli, F. (1997). Adaptive selection of image classifiers. *ICIAP '97: Proc. of the 9th International Conference on Image Analysis and Processing-Volume I* (pp. 38–45). London, UK: Springer-Verlag.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, *10*, 447–474.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*, 302–320.

Goodman, J. (2001). *A bit of progress in language modeling* (Technical Report MSR-TR-2001-72). Microsoft Research Technical Report.

Günter, S., & Bunke, H. (2002). Generating classifier ensembles from multiple prototypes and its application to handwriting recognition. *MCS '02: Proc. of the Third International Workshop on Multiple Classifier Systems* (pp. 179–188). London, UK: Springer-Verlag.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transaction on Pattern Analyis and Machine Intelligence*, *12*, 993–1001.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning*. Springer.

Heskes, T. (1997). Balancing between bagging and bumping. *Advances in Neural Information Processing Systems* (p. 466). The MIT Press.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 832–844.

Howe, N. R., & Cardie, C. (1999). *Weighting unusual feature types* (Technical Report TR99-1375). Cornell University.

Islam, M., Yao, X., & Murase, K. (2003). A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks, 14(4):820– 834, July 2003.*, *14*, 820–834.

Jacquemont, S., Jacquenet, F., & Sebban, M. (2005). Constrained sequence mining based on probabilistic finite state. *Proc. of the Workshop on Mining Graphs, Trees and Structured Data at ECML/PKDD*.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. *Proc. of the 15 th International Conference on Uncertainty in Artificial Intelligence* (pp. 338–345).

Kearns, M., & Mansour, Y. (1996). On the boosting ability of top-down decision tree learning algorithms. *STOC '96: Proc. of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 459–468). New York, NY, USA: ACM Press.

Kearns, M. J., & Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. M.I.T. Press.

Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Transaction on Pattern Analysis Machine Intelligence*, *20*, 226–239.

Kivinen, J., & Warmuth, M. (1999a). Boosting as entropy projection. *Proc. of the 12 th International Conference on Computational Learning Theory* (pp. 134–144).

Kivinen, J., & Warmuth, M. K. (1999b). Boosting as entropy projection. *Computational Learing Theory* (pp. 134–144).

Kohavi, R. (1994). Feature subset selection as search with probabilistic estimates. *AAAI Fall Symposium on Relevance* (pp. 122–126).

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Machine Learning: Proc. of the Thirteenth International Conference* (pp. 275–283). Morgan Kaufmann.

Koltchinskii, V., & Panchenko, D. (2002). Empirical margin distributions and

bounding the generalization error of combined classifiers. *Annals of Statistics*, *30*, 1–50.

Kong, E. B., & Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance. *International Conference on Machine Learning* (pp. 313–321).

Krause, N., & Singer, Y. (2004). Leveraging the margin more carefully. *Machine Learning, Proc. of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. ACM.

Kuncheva, L., & Kounchev, R. K. (2002). Generating classifier outputs of fixed accuracy and diversity. *Pattern Recognition Letters*, *23*, 593–600.

Kuncheva, L., Skurichina, M., & Duin, R. (2002). An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion*, *3*, 245–258.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, *51*, 181–207.

Lane, T., & Brodley, C. E. (2000). Data reduction techniques for instance-based learning from human/computer interface data. *Proc. 17th International Conf. on Machine Learning* (pp. 519–526). Morgan Kaufmann, San Francisco, CA.

Lang, K., Pearlmutter, B., & Price, R. (1998). Results of the abbadingo one DFA learning competition. *4^{th} Int. Coll. on Grammatical Inference* (pp. 1–12).

Lazarevic, A., & Obradovic, Z. (2001). Adaptive boosting techniques in heterogeneous and spatial databases. *Intelligent Data Analysis*, *5*, 285–308.

Lecce, V., Dimauro, G., Guerriero, A., Impedovo, S., Pirlo, G., & Salzo, A. (2000). Classifier combination: The role of a-priori knowledge. *Proc. of the seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, Netherlands* (pp. 143–152).

Levenshtein, V. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, *10*, 707–710. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).

Littlestone, N., & Warmuth, M. K. (1989). The weighted majority algorithm. *IEEE Symposium on Foundations of Computer Science* (pp. 256–261).

Liu, H., & Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution. *International Conference on Machine Learning* (pp. 319–327).

Maclin, R. (1998). Boosting classifiers regionally. *AAAI '98/IAAI '98: Proc. of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence* (pp. 700–705). Menlo Park, CA, USA: American Association for Artificial Intelligence.

Maclin, R., & Opitz, D. (1997). An empirical evaluation of bagging and boosting. *Proc. of the Fourteenth National Conference on Artificial Intelligence* (pp. 546–551).

Margineantu, D. D., & Dietterich, T. G. (1997). Pruning adaptive boosting. *Proc. 14th International Conference on Machine Learning* (pp. 211–218). Morgan Kaufmann.

Mary, J. (2005). *Etude de l'apprentissage actif : Application à la conduite d'expériences*. Doctoral dissertation, Université d'Orsay.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems 12, pages 512–518*. MIT Press.

Meir, R., & Raetsch, G. (2003). An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning* (pp. 119–184). LNCS 2600.

Merz, C. J. (1996). Dynamical selection of learning algorithms. In D. Fisher et H.-J. Lenz (Eds.), *Learning from data: Artificial intelligence and statistics*. New York, NY: Springer Verlag.

Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Machine Learning*, *36*, 33–58.

Mico, L., & Oncina, J. (1998). Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recogn. Lett.*, *19*, 351–356.

Mitchell, T. M. (1990). The need for biases in learning generalizations. In J. W. Shavlik et T. G. Dietterich (Eds.), *Readings in machine learning*, 184–191. San Mateo, CA: Kaufmann.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Nock, R., & Sebban, M. (2001). A bayesian boosting theorem. *International Journal of Pattern Recognition Letters*, *22*, 413–419.

Oncina, J., & García, P. (1992). *Inferring regular languages in polynomial update time*, vol. 1 of *Machine Perception and Artificial Intelligence*, 49–61. World Scientific.

Oncina, J., & Sebban, M. (2006). Learning stochastic edit distance: application in handwritten character recognition. *Journal of Pattern Recognition*. (to appear).

Opitz, D. W., & Shavlik, J. W. (1996). Actively searching for an effective neural-network ensemble. *Connection Science*, *8*, 337–353.

Perrone, M. P., & Cooper, L. N. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone (Ed.), *Neural networks for speech and image processing*, 126–142. Chapman-Hall.

Pitt, L. (1989). Inductive inference, DFA's, and computational complexity. In *Analogical and inductive inference*, no. 397 in LNAI, 18–44. Berlin, Heidelberg: Springer.

Quinlan, J. R. (1988). Simplifying decision trees. In B. Gaines et J. Boose (Eds.), *Knowledge acquisition for knowledge-based systems*, 239–252. London: Academic Press.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Quinlan, J. R. (1996). Bagging, boosting, and c4.5. *AAAI/IAAI, Vol. 1* (pp. 725–730).

Ricci, F., & Aha, D. W. (1998). Error-correcting output codes for local learners. *ECML '98: Proc. of the 10th European Conference on Machine Learning* (pp. 280–291). London, UK: Springer-Verlag.

Ristad, E. S., & Yianilos, P. N. (1997). Learning string edit distance. *ICML '97: Proc. of the Fourteenth International Conference on Machine Learning* (pp. 287–295). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, 318–362.

Rätsch, G., Onoda, T., & Müller, K. R. (1999). Regularizing adaboost. *Proc. of the 1998 conference on Advances in neural information processing systems II* (pp. 564–570). Cambridge, MA, USA: MIT Press.

Rätsch, G., & Warmuth, M. (2002). Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, *6*.

Rätsch, G., Warmuth, M. K., Mika, S., Onoda, T., Lemm, S., & Müller, K.-R. (2000). Barrier boosting. *COLT '00: Proc. of the Thirteenth Annual Conference on Computational Learning Theory* (pp. 170–179). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Sanderson, C., Bengio, S., Bourlard, H., Mariethoz, J., Collobert, R., BenZeghiba, M. F., Cardinaux, F., & Marcel, S. (2003). Speech & face based biometric authentication at idiap. *Proc. IEEE International Conference on Multimedia & Expo (ICME)*.

Schapire, R. (2001). The boosting approach to machine learning: An overview. *MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA*.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Schapire, R. E. (1997). Using output codes to boost multiclass learning problems. *Proc. 14th International Conference on Machine Learning* (pp. 313–321). Morgan Kaufmann.

Schapire, R. E. (1999). A brief introduction to boosting. *IJCAI '99: Proc. of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 1401–1406). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. *Proc. 14th International Conference on Machine Learning* (pp. 322–330). Morgan Kaufmann.

Schapire, R. E., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, *37*, 297–336.

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, *39*, 135–168.

Sebban, M., & Janodet, J. (2003). On state merging in grammatical: a statistical approach for dealing with noisy data. *Proc. of the 20 th International Conference on Machine Learning* (pp. 688–695). See also CAp'03.

Sebban, M., Janodet, J., Suchier, H., & Nock, R. (2004). Boosting grammatical inference with confidence oracles. *International Conference on Machine Learning (ICML)* (pp. 425–432).

Sebban, M., Nock, R., & Lallich, S. (2001). Boosting neighborhood-based classifiers. *ICML '01: Proc. of the Eighteenth International Conference on Machine Learning* (pp. 505–512). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Sebban, M., Nock, R., & Lallich, S. (2003). Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. *Journal of Machine Learning Research*, 863–885.

Sebban, M., & Suchier, H. (2003). On boosting improvement: Error reduction and convergence speed-up. *14th European Conference on Machine Learning (ECML)* (pp. 349–360).

Seeger, M. (2000). Learning with labeled and unlabeled data. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

Servedio, R. A. (2001). Smooth boosting and learning with malicious noise. *14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 2001, Proc.* (pp. 473–489). Springer, Berlin.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423,623–656.

Shipp, C. A., & Kuncheva, L. (2002). Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, *3*, 135–148.

Solla, S. A., Leen, T. K., & Müller, K.-R. (Eds.). (2000). *Advances in neural*

information processing systems 12, [nips conference, denver, colorado, usa, november 29 - december 4, 1999]. The MIT Press.

Thollard, F., Dupont, P., & de la Higuera, C. (2000). Probabilistic DFA inference using Kullback-Leibler divergence and minimality. *Proc. 17th International Conf. on Machine Learning* (pp. 975–982). Morgan Kaufmann, San Francisco, CA.

Ting, K. M. (2000). An empirical study of metaCost using boosting algorithms. *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2, 2000, Proc.* (pp. 413–425). Springer, Berlin.

Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, *10*, 271–289.

Tsoumakas, G., Katakis, I., & Vlahavas, I. P. (2004). Effective voting of heterogeneous classifiers. *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings* (pp. 465–476).

Tsymbal, A., Puuronen, S., & Terziyan, V. Y. (1999). Arbiter meta-learning with dynamic selection of classifiers and its experimental investigation. *Advances in Databases and Information Systems* (pp. 205–217).

Valiant, L. G. (1984). A theory of the learnable. *STOC '84: Proc. of the sixteenth annual ACM symposium on Theory of computing* (pp. 436–445). New York, NY, USA: ACM Press.

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, *16*, 264–280.

Vapnik, V. N. (1998). *The statistical learning theory*. Wiley. VAP v 98:1 1.Ex.

Verlinde, P., Chollet, G., & Acheroy, M. (2000). Multi-modal identity verification using expert fusion. *Information Fusion*, *1*, 17–33.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Conference on Computer Vision and Pattern Recognition* (p. 551). IEEE Computer Society.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, *21*, 168–173.

Wilson, D., & Martinez, T. (1997). Instance pruning techniques. *Proc. of the 14 th International Conference on Machine Learning* (pp. 404–411).

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*, 257–286.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Table des figures

Liste des tableaux